

BASIS 108

Betriebsanleitung

HINWEIS:

Änderungen bedingt durch technischen Fortschritt bleiben vorbehalten.

INHALTSVERZEICHNIS

Kapitel 1

Allgemeine Beschreibung und Inbetriebnahme

Kapitel 2

Software

Kapitel 3

Zugriff zur Hardware

Kapitel 4

Der Monitor

Kapitel 5

Der Speicher

Kapitel 6

Ein-/Ausgabe

ANHANG

Vorwort

In diesem Handbuch finden Sie neben einer Reihe sehr einfacher Hinweise für den Umgang mit Ihrem Computer eine Vielzahl von Hinweisen, die vor allem für den fortgeschrittenen Programmierer von Interesse sind.
Für den Anfänger ist dieses Buch in weiten Passagen wohl kaum verständlich. Deswegen sollte er sich auch zunächst mit Einführungen in die Programmierung und Arbeitsweise eines Computers beschäftigen, ehe er intensiver mit diesem Handbuch arbeitet. Er sollte aber die Kapitel 1 und 2, sowie Teile des Anhangs, die ihn evtl. betreffen, auch wenn Ihm andere Programmierhandbücher zur Verfügung stehen, zunächst lesen.
Zum Teil werden hier auch Möglichkeiten aufgezeigt, die aus der Kompatibilität des BASIS 108 mit dem Apple II resultieren. Möglichkeiten also, die z.B. Anwender des UCSD p-Systems IV.0 kaum interessieren.

Ein Handbuch wird geschrieben für den Anwender, deshalb hier zum Schluß die Bitte an Sie: Wenn Sie Kritik und Anregungen haben, so teilen Sie uns diese mit, damit wir sie bei der nächsten Auflage berücksichtigen können.

Wir wünschen Ihnen erfolgreiche Arbeit mit Ihrem BASIS 108.

KAPITEL 1

INHALTSVERZEICHNIS

Das BASIS 108 Computersystem und Datensichtgeräte

Ihr BASIS 108 Computersystem besteht aus folgenden Teilen:

1. Der Zentraleinheit mit oder ohne eingebauten Diskettenlaufwerken,
2. der Tastatur,
3. dem Netzanschlußkabel,
4. der Diskette ZAP:, auf der Rückseite befindet sich Volume UT108:,
5. und diesem Handbuch.

Bewahren Sie das Verpackungsmaterial bitte auf, falls Sie das System einmal transportieren wollen, bietet es guten Schutz vor Beschädigung des Computers.
Zum Betrieb des Systems benötigen Sie noch einen Bildschirm (Datensichtgerät) oder, falls Ihnen 40 Zeichen/ Zeile genügen, ein Fernsehgerät mit Video-Eingang. (Mehr als 40 Zeichen/ Zeile kann ein normales Fernsehgerät nicht sauber darstellen).
Für den Anschluß eines Fernsehgerätes ohne Videoeingang s. Anhang F.
Sollten Sie großen Wert auf gute Farbausgabe legen, dann benötigen Sie einen hochauflösenden RGB-Monitor. Ihr BASIS Vertriebspartner wird Sie auch in dieser Angelegenheit beraten.

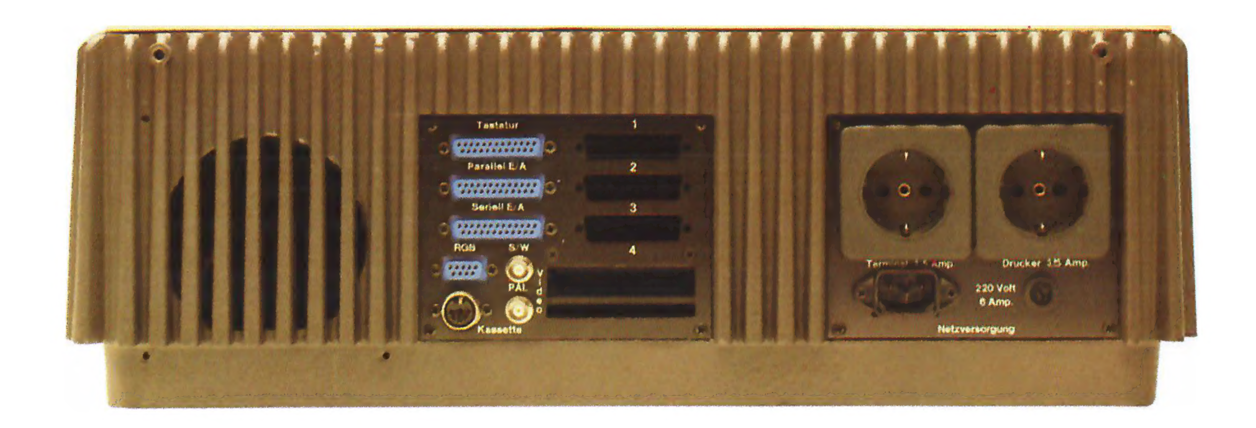
Anschluß der Kabel

Wenn Sie ein BASIS 108 System ohne Diskettenlaufwerke erworben haben und die ersten Schritte mit Ihrem eigenen Computer per Kassettenrekorder zurücklegen wollen, dann schließen Sie Ihren Kassettenrekorder an die dafür vorgesehene DIN-Buchse auf der Rückseite des BASIS 108 an, weiteres s. Anhang G.

Wichtig: Fragen Sie Ihren BASIS Vertriebspartner nach dem richtigen Monitor-ROM zum Laden des Betriebssystems mit Kassettenrekorder.

Haben Sie Ihr BASIS 108 Computersystem mit Diskettenlaufwerken erworben, um damit eine Arbeitserleichterung bei Ihren täglichen Routinearbeiten zu erzielen, so haben Sie keinerlei Anschlußarbeiten.
Ein eigenes Laufwerk sollten Sie entsprechend der Anleitung Seite 11 einbauen.
Die Steckdosen auf der Rückseite sind für den Bildschirm und Drucker vorgesehen. Verbinden Sie also alle Einheiten miteinander, indem Sie das Netzkabel des Bildschirms und des Druckers in die dafür vorgesehenen Steckdosen auf der Rückseite des BASIS 108 einstecken.
Diese beiden Steckdosen werden über den zentralen Netzschalter des Systems geschaltet.

Wichtig: Bitte die Steckdosen nur für Drucker und Bildschirm benutzen, nicht für Staubsauger etc.



Rückseite

Verbinden Sie den Bildschirm oder das Fernsehgerät durch ein Video-Kabel mit dem RGB, S/W-Video oder PAL-Video Ausgang des Systems.
Stecken Sie den Stecker der Tastatur in den dafür vorgesehenen Buchsenstecker auf der Rückseite des Gerätes.
In der Betriebsanweisung Ihres Druckers finden Sie Angaben darüber, ob er über eine serielle oder parallele Schnittstelle verfügt. Entsprechend können Sie die Verbindung zum BASIS 108 herstellen, indem Sie das Datenkabel zur Rückseite führen und es in die infrage kommende Steckleiste stecken.
Verbinden Sie nun das System über das Netzkabel mit der nächsten Steckdose und vergewissern Sie Sich noch einmal, ob alle Geräte richtig verbunden sind. Jetzt schalten Sie den Netzschalter an der unteren linken Seite der Front des BASIS 108 ein.
Die rote Lampe leuchtet auf, der eingebaute Lautsprecher piept kurz und das linke Diskettenlaufwerk läuft an.
Auf dem Bildschirm erscheint die Meldung:

B A S I S 1 0 8

Da Sie mehrere Betriebssysteme und Zusatzgeräte verwenden können, ist es notwendig, die grundsätzliche Arbeit mit Ihrem Computer in einem gesonderten Kapitel zu besprechen.
Wenn Sie nicht mehr über Ihren BASIS 108 wissen möchten, dann lesen Sie bitte Kapitel 2.
Falls Sie aber Ihren persönlichen Computer näher kennenlernen möchten, dann lesen Sie weiter.

Belegung der Pins bei den Steckleisten der Rückseite

Auf der nächsten Seite finden Sie die Zeichnung mit der Rückseite. Hier sind die entsprechenden Pins der Steckleisten bezeichnet. Die Bedeutung der Zeichen ergibt sich zum Teil aus der Beschriftung.
Die Bezeichnungen D0 - D7 sind von der Tastatur her Dateneingänge, bei der parallelen Schnittstelle die Ausgänge der Druckzeichen.
Die Bezeichnung GND bedeutet Gerätemasse.

SM ist dagegen die Signalmasse.
Ausgang sind die Signale: RTS, DTR, R, G, B.
Eingang sind die Signale: CTS, DSR, DCD, PC, DI, AC.

Die Abkürzungen der Signale bei der seriellen Schnittstelle entnehmen Sie bitte im Anhang dem Datenblatt des 6551.

ST Strobe ist ein negatives Signal mit 1 Mikrosekunde Dauer .
AC Ist ein negatives Antwortsignal mit 1 Mikrosekunde Dauer (Acknowledge).
PC (Printer Connect) ist auf 0 gezogen, wenn der Drucker eingeschaltet ist.

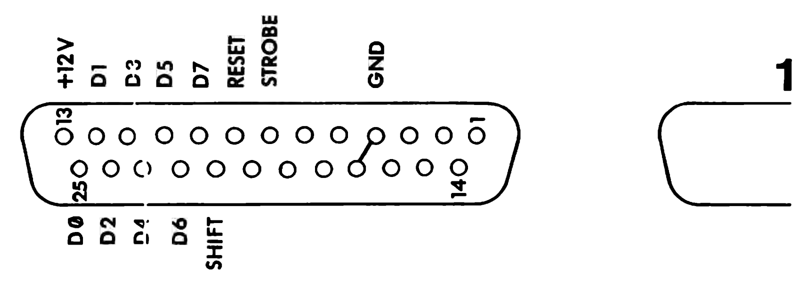
Die beiden 12 V Anschlüsse der seriellen Schnittstelle sind durch Widerstände von 1 kOhm geschützt.
Ist der Eingang CTS inaktiv, dann erfolgt keine Sendung.

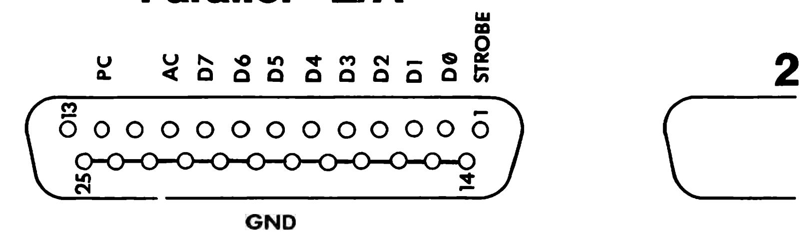
Öffnen des BASIS 108

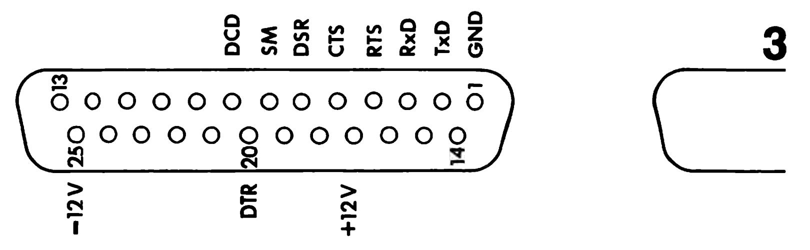
Wichtig: Bevor Sie das System öffnen, ziehen Sie bitte den Netzstecker aus der Steckdose

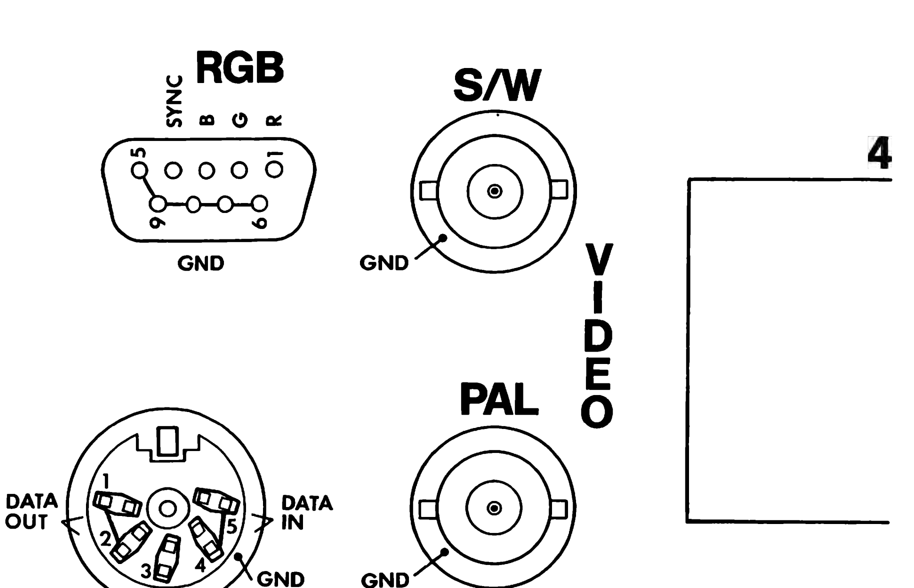
Das BASIS 108 System besteht aus einem Aluminium-Gußgehäuse mit dem eingebauten Netzteil und der Hauptplatine. In der Front des Gehäuses sind Öffnungen zum Einbau von zwei Diskettenlaufwerken, die durch Blindabdeckungen verschlossen sind, wenn keine Laufwerke eingebaut wurden. Montagebleche und Befestigungsschrauben für Diskettenlaufwerke sind aber in jedem Fall vorhanden, siehe S. 12.
In der Mitte finden Sie neben den schon belegten Buchsensteckern für die Tastatur und die serielle sowie parallele Schnittstelle noch drei weitere Montageplätze für DP-25 Buchsenstecker.
Darunter befindet sich neben den Anschlußbuchsen für einen RGB-Monitor, PAL-Video Fernseher und S/W Bildschirm (BNC-Buchse) ein Durchbruch zum direkten Herausführen von Flachbandkabeln bis zu einer Breite von 50 Adern.

Das Gehäuse besteht aus zwei Teilen: dem Unterteil mit der hochgezogenen Rückwand und dem Deckel. Der Deckel wird an der Rückwand des Unterteils von zwei Metallstiften gehalten und durch zwei Schrauben, die sich im forderen Bereich des Unterteils befinden, gesichert.
Heben Sie das System an und lösen Sie die Schrauben mit einem stabilen Schraubenzieher. Ziehen Sie nun das Oberteil nach vorne ab.


Tastatur
+12V
D1
D3
D5
D7
RESET
STROBE
GND
13
1
25
14
DØ
D2
D4
D6
SHIFT
1


Parallel · E/A
PC
AC
D7
D6
D5
D4
D3
D2
D1
DØ
STROBE
13
1
25
14
GND
2


Seriell · E/A
DCD
SM
DSR
CTS
RTS
RxD
TxD
GND
13
1
25
20
14
−12V
DTR
+12V
3


RGB
SYNC
B
G
R
5
1
9
6
GND
S/W
GND
VIDEO
PAL
GND
DATA OUT
DATA IN
GND
Kassette
4

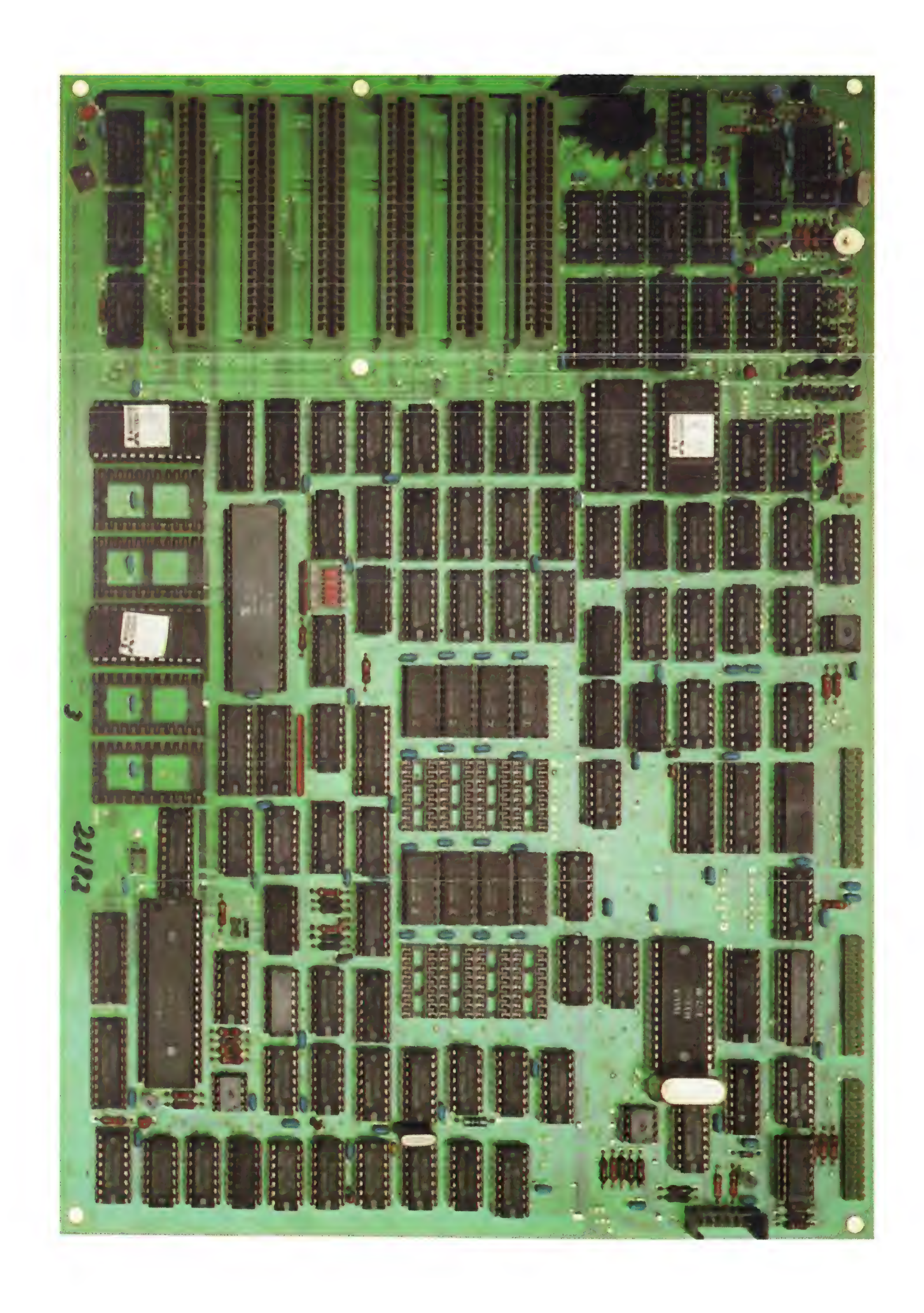


Innenansicht

Die Hauptplatine

Die große Leiterplatte ist der eigentliche Computer. Auf ihr sind ca. 130 hochintegrierte Schaltkreise, ICs, untergebracht, die die elektrische Verbindung zwischen den zwei Mikroprozessoren (6502 und Z-80), den Speicherbausteinen RAM (Random Access Memory) und ROM (Read only Memory) und den Ein-/Ausgabebausteinen herstellen.
Auf der linken Seite der Platine befinden sich sechs 50-polige Slots (Buchsenleisten), von 2 bis 7 numeriert, in die Systemerweiterungen wie z.B. Steuereinheiten für Diskettenlaufwerke, serielle und parallele Schnittstellenkarten für weitere Drucker oder Hauptspeichererweiterungen eingesetzt werden können. Wenn Ihr System mit Diskettenlaufwerken ausgestattet ist, dann steckt in dem Steckplatz 6 die Steuereinheit, der Controller. Dieser Controller kann bis zu zwei Diskettenlaufwerke kontrollieren.
Hinten rechts befinden sich drei Stiftleisten mit je 20 Stiften, von denen aus Flachbandkabel zu den Buchsensteckern auf der Rückwand des Systems führen, für die Tastatur, sowie für einen parallel und einen seriell anzusteuernden Drucker.
Hinten in der Mitte der linken Seite ist eine Stiftleiste mit 10 Stiften. Hierüber wird das RGB-Signal über ein Flachbandkabel auf den entsprechenden Stecker auf der Rückseite gegeben. Rechts daneben befindet sich der schwarz/weiß Video-Ausgang (S/W-Video). Der Ausgang für PAL-Video bzw. den Anschluß eines UHF-Modulators ist die Steckleiste mit den vier Stiften in der linken oberen Ecke der Platine.
Die Farbqualität bei Farbausgabe läßt sich über den Trimmkondensator, links oben, mit Hilfe eines kleinen Schraubenziehers einstellen. Die Intensität des S/W-Videosignales läßt sich über das rechts in der Nähe des Trimmkondensators stehende Potentiometer regeln.
Der auf der rechten Seite der Platine angebrachte Stecker führt ein Verbindungskabel zum Lautsprecher und zum Kassettenrekorder-Anschluß.
Der große Stecker direkt hinter der Buchsenleiste 7 verbindet über ein Anschlußkabel das Netzteil mit der Hauptplatine.

Etwa in der Mitte der Platine sind die Hauptspeicherbausteine (RAMs) angeordnet. In der Grundausstattung des BASIS 108 befinden sich 8 IC's mit je 64 KBit in den eingelöteten Sockeln. Weitere 8 Bausteine können durch einfaches Einsetzen in die dafür vorgesehenen Steckplätze nachgerüstet werden und erweitern dann den Hauptspeicher auf eine Kapazität von insgesamt 128 KByte.
Da die verwendeten 8 Bit Mikroprozessoren 6502 und Z-80 nur einen Speicheradressraum von 65 536 Bytes (64 KBytes) ansprechen können, benötigen Sie zum Adressieren des Gesamtspeicherraumes von 2x 65 536 Bytes ein spezielles Programm, das Sie von Ihrem BASIS Vertriebspartner beziehen können.
In der ersten Reihe auf der Platine sind sechs Sockel angeordnet, von denen zwei Sockel durch integrierte Bausteine belegt sind. Diese Sockel sind für Festwertspeicher (ROMs) reserviert. Sie können Programme oder Programmiersprachen aufnehmen, die im Augenblick des Einschaltens des BASIS 108 verfügbar werden. Eines dieser Programme ist schon in dem linken Baustein vorhanden; der BASIS 108 System-Monitor. Mit Hilfe dieses Monitors (Programmes) wird nach dem Einschalten des Systems das linke Diskettenlaufwerk (Laufwerk 1) angesteuert, hierzu weiteres in Kapitel 2 und 4.
Ist kein Laufwerk eingebaut, können Sie Programme vom Kassettenrekorder einlesen, wenn in Ihrem BASIS 108 ein 40 Spalten Monitor-ROM eingebaut ist.
Weiteres hierzu siehe Anhang G.



Hauptplatine

Die beiden Schaltungsbrücken in der Nähe des 6502 sind zur Umschaltung zwischen ROM- und EPROM-Bestückung. Im Lieferzustand befinden sich die beiden Jumper (Kurzschlußbrücken) in der Position EPROM. In diesem Zustand sind das eingesetzte BASIS-Monitor-EPROM und das "Dummy"-EPROM aktiv geschaltet. Soll ein kompletter Satz EPROMs vom Typ 2716 installiert werden, wird die Jumper-Stellung nicht verändert.
Bei Einsatz der ROM-Bestückung (original Applesoft- oder Integer-ROMs) müssen beide Jumper in die entgegengesetzte Position.
Die Beschreibung der Stellung des Dip-Schalters über dem Z-80 finden Sie im Anhang bei der Beschreibung des Z-80 Teiles.

Der Handregleranschluß

Links hinter der Buchsenleiste 7 befindet sich ein nicht mit einem IC bestückter Sockel. Dieser Sockel dient der Aufnahme eines Steckers von Handreglern (Game Paddle oder Joystic). Die Kabel müssen nach links aussen zeigen. Entsprechende Spielprogramme fordern Sie auf, die Handregler anzuschließen.

Im folgenden sind die Handregleranschlußbelegung und die Beschreibung der Spielanschlußsignale wiedergegeben.

Handregleranschlußbelegung

Signal	Pin	Pin	Signal
+5V	1	16	NC
SW0	2	15	AN0
SW1	3	14	AN1
SW2	4	13	AN2
C040 STB	5	12	AN3
PDL0	6	11	PDL3
PDL2	7	10	PDL1
GND	8	9	NC

Beschreibung der Handregleranschlußsignale

Anschluß	Name	Beschreibung
1	+5V	+5 V Stromversorgung, max. 100 mA.
2-4	SW0-SW2	Ein-Bit-Eingänge (Drucktasten). Es sind Standart-TTL-Eingänge der 74LS-Serie.
5	C040 STB	Der Impulsausgang ist ein Standart-TTL 74LS-Ausgang. Dieser Anschluß liegt normalerweise an +5 V und geht beim Zugriff auf eine Adresse von $C040 bis $C04F für die Dauer von 0.4 Mikrosekunden in Phase Φ_0 auf logisch 0.

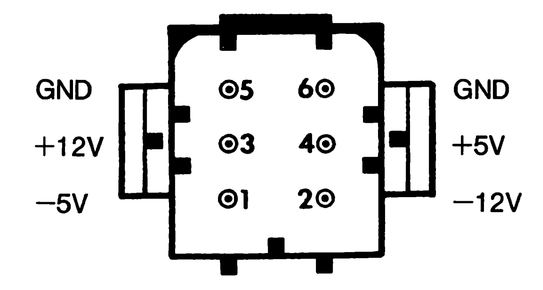
Anschluß	Name	Beschreibung (Forts.)
6,7,10,11	PDL0-PDL3	Spielsteuereingänge. Diese Analogeingänge sollten mit 150 kOhm-Regelwiderständen an +5 V angeschlossen werden.
8	GND	Elektrische Masse des Systems: 0 V.
12-15	AN0-AN3	Signal-Ausgänge (Annunciator). Diese Standartausgänge der TTL 74LS-Serie sollten gepuffert werden, falls sie andere als TTL-Eingänge treiben sollen.
9,16	NC	Kein Anschluß.

Die Stromversorgung

Das Metallgehäuse auf der linken Seite neben der Hauptplatine ist das Netzteil. Es liefert vier Spannungen:

+5 Volt,
-5 Volt,
+12 Volt,
-12 Volt.

Die Pinbelegung entnehmen Sie der Abbildung:



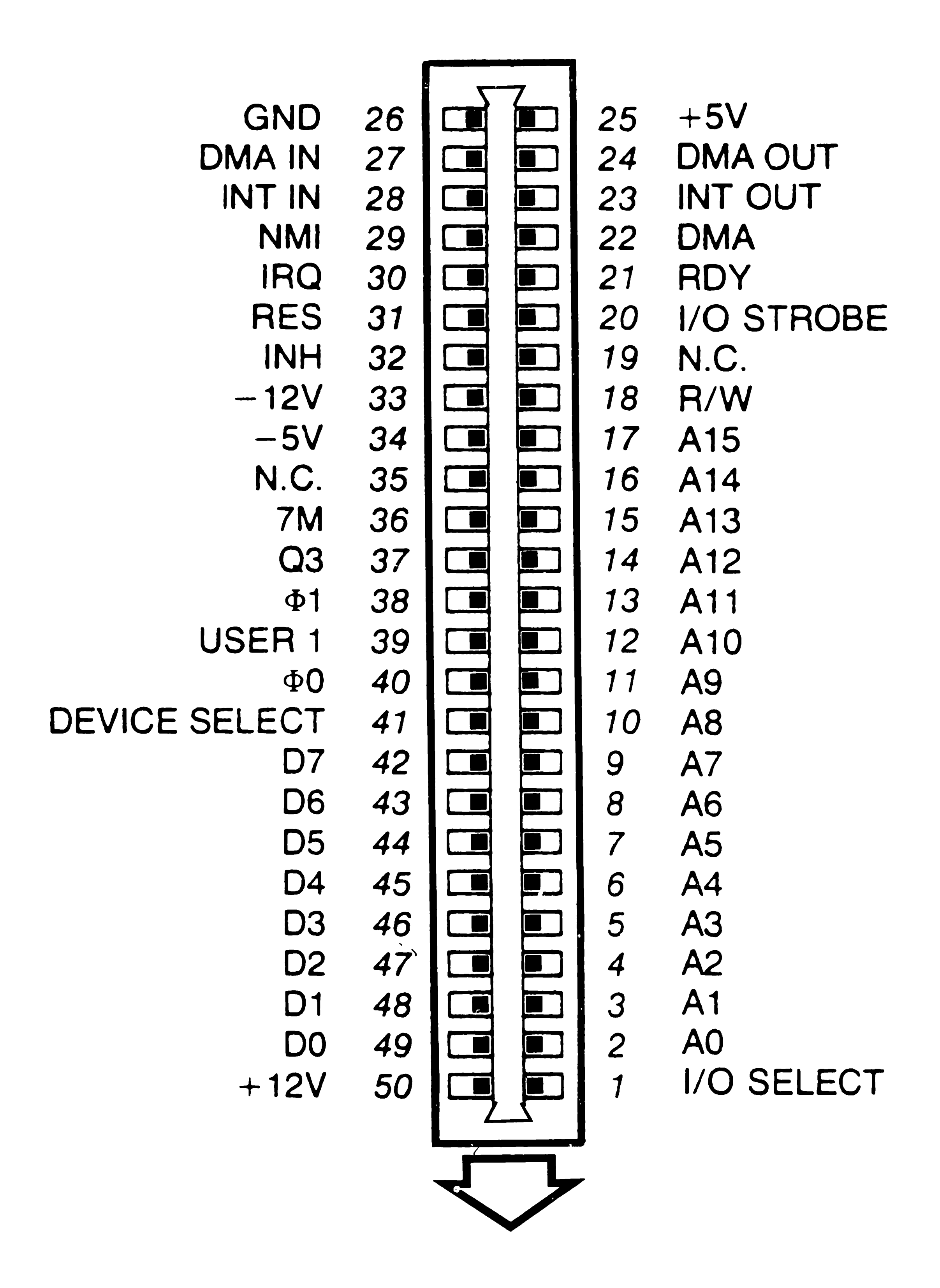
Das getaktete Netzteil wurde mit einer Schutzeinrichtung versehen, damit keine Überlastung auftreten kann. Die Eingangsseite kann an 110 Volt bis 250 Volt angeschlosen werden, bei 110 Volt muß im Netzteil ein Stecker umgesteckt werden, und ist über ein Kabel mit dem an der Rückseite des Systems angebrachten Netzfilter verbunden.

Wichtig: Das Netzteil nicht öffnen! Lebensgefährliche Spannungen!

Pinbelegung der Slots

Im folgenden ist die Pinbelegung der Slots aufgeführt. Die Zeichnung finden Sie auf der nächsten Seite. Die aufgeführten Zahlen mit einem $-Zeichen sind Hexadezimalzahlen. Bitte sehen Sie hierzu in den Anhang H und in die Kapitel Monitor ff.

Anschluß	Name	Beschreibung
1	$\overline{\text{I/O SELECT}}$	Diese Leitung liegt normalerweise auf +5 V. Wenn der Mikroprozessor auf Seite $Cn zugreift (wobei n die Slotnummer ist), sinkt die Spannung auf logisch 0 ab. Dieses Signal wird während Φ_1 aktiv und treibt 10 LS-TTL-Lasten.
2-17	A0-A15	Der gepufferte Adressbus. Die Adressen werden in Φ_1 gültig und bleiben es in Φ_0 Jede dieser Leitungen treibt 5 LS-TTL-Lasten.
18	R/$\overline{\text{W}}$	Gepuffertes Lese-/Schreib-Signal (Read/Write). Dieses Signal ist zur selben Zeit gültig wie der Adressbus und geht auf +5 V in einem Lese- und auf logisch 0 in einem Schreibvorgang. Diese Leitung kann 2 LS-TTL-Lasten versorgen.
20	$\overline{\text{I/O STROBE}}$	Diese Leitung treibt 4 LS-TTL-Lasten und geht während Φ_0 auf 0, wenn der Adressbus eine Adresse zwischen $C800 und $CFFF enthält.
21	RDY	Der RDY-Eingang des 6502-Mikroprozessors. Wird diese Leitung während Φ_1 auf 0 gezogen, so stopt der Mikroprozessor und hält die aktuelle Adresse im Adressbus fest.
22	$\overline{\text{DMA}}$	Wird dieser Anschluß auf logisch 0 gelegt, so wird der Adressbus gesperrt und der Mikroprozessor gestopt. Diese Leitung wird durch einen 1 KOhm Widerstand auf +5 V gehalten.
23	INT OUT	Daisy-Chain Interrupt-Ausgang zu Geräten niedriger Priorität. Dieser Anschluß wird normalerweise mit Pin 28 (INT IN) verbunden. INT OUT 7 führt zum Z-80-Teil.
24	DMA OUT	Daisy-Chain DMA-Ausgang zu Geräten niedrigerer Priorität. Dieser Anschluß wird normalerweise mit Pin 22 (DMA IN) verbunden. DMA OUT 7 führt zum Z-80-Teil.



Pinbelegung der Slots

Anschluß	Name	Beschreibung
25	+5 V	+5 V Stromversorgung. Für alle Peripheriekarten stehen insgesamt 3 A zur Verfügung.
26	GND	Elektrische Masse des Systems.
27	DMA IN	Daisy-Chain DMA-Eingang von Geräten höherer Priorität. Gewöhnlich mit Anschluß 24 (DMA OUT) verbunden.
28	INT IN	Daisy-Chain Interrupt-Eingang von Geräten höherer Priorität. Gewöhnlich mit Anschluß 23 (INT OUT) verbunden. INT IN von Slot 2 kommt von der seriellen Schnittstelle der Tastatur.
29	$\overline{NMI}$	Nicht maskierbarer Interrupt (hardwaremäßiges Einschieben eines speziellen Unterprogrammes). Wenn diese Leitung auf 0 gezogen wird, beginnt der BASIS 108 einen Interrupt-Ablauf und springt dann zu einem Interrupt-Behandlungs-Programm auf Adresse $3FB.
30	$\overline{IRQ}$	Maskierbarer Interrupt (Interrupt ReQuest). Wenn diese Leitung auf logisch 0 liegt und das I-Bit des 6502-Mikroprozessors (Interrupt Sperre) nicht gesetzt ist, beginnt der BASIS 108 einen Interrupt Ablauf und springt zu dem Interrupt-Behandlungsprogramm, dessen Adresse in den Speicherzellen $3FE und $ 3FF zu finden sind.
31	$\overline{RES}$	Wird dieser Anschluß auf logisch 0 gelegt, so beginnt der Mikroprozessor einen (RESET)-Ablauf.
32	$\overline{INH}$	Wenn diese Leitung auf 0 gezogen wird, wird der obere 12 K Adressraum auf der Platine abgeschaltet. Diese Leitung wird durch einen 1 kOhm Widerstand auf +5 V gehalten.
33	-12 V	-12 V Spannungsversorgung. Der Maximalstrom beträgt 0,5 A für alle Peripheriekarten zusammen.
34	-5 V	-5 V Spannungsversorgung. Der maximal zulässige Strom beträgt für alle Peripheriekarten zusammen 0,5 A.

Anschluß	Name	Beschreibung
35	darf nicht beschaltet werden.	
36	7M	7 MHz Takt. Diese Leitung treibt zwei LS-TTL-Lasten.
37	Q3	Asymmetrischer 2 MHz Takt. Dieser Anschluß treibt zwei LS-TTL-Lasten.
38	Φ_1	Phase 1-Takt des Mikroprozessors. Dieser Anschluß kann zwei LS-TTL-Lasten versorgen.
39	USER 1	Wenn diese Leitung auf 0 gezogen wird, ist der $Cxxx-Bereich unterbrochen.
40	Φ_0	Phase 0-Takt des Mikroprozessors. Dieser Anschluß kann zwei LS-TTL-Lasten versorgen.
41	$\overline{\text{DEVICE SELECT}}$	Leitung wird auf jedem Peripherieanschluß aktiv (logisch 0), wenn der Adressbus eine Adresse zwischen $C0n0 und $C0nF gespeichert hat, wobei n die um $8 erhöhte Slotnummer angibt. Diese Leitung treibt 10 LS-TTL-Lasten.
42-49	D0-D7	In zwei Richtungen gepufferter Datenbus. Die Dateninformation auf dieser Leitung liegt mindestens 300 ns in Phase 0 beim Schreiben und sollte beim Lesen nicht länger als 100 ns vor dem Ende von Φ_0 erhalten bleiben.
50	+12 V	+12 V Stormversorgung. Bis zu 2,5 A können insgesamt an alle Peripheriekarten abgegeben werden.

Die Diskettenlaufwerke

Die Verwendung von Diskettenlaufwerken in Verbindung mit dem BASIS 108 System ist weitaus schneller und einfacher als die Verwendung eines Kassettenrekorders.
Jedes BASIS 108 System ist mit Halteblechen für zwei Diskettenlaufwerke ausgerüstet. Wenn keine Laufwerke eingebaut sind, befinden sich die Befestigungsschrauben für die Laufwerke in einer kleinen Plastiktüte an den Halteblechen.
Falls Sie Diskettenlaufwerke nachträglich montieren wollen, dann schrauben Sie nach Abnehmen des Gehäusedeckels die Haltebleche von dem Gehäuseboden ab.

Wichtig: Vergewissern Sie sich, ob auch der Netzstecker gezogen ist und die kleine rote Kontroll-Lampe auf der Hauptplatine aus ist.

Montieren Sie jetzt mit Hilfe der mitgelieferten Schrauben die Haltebleche an die Laufwerke und setzen anschließend die komplett montierten Einheiten wieder an ihren Platz zurück. Bevor Sie die Bleche am Gehäuseboden fest montieren, legen Sie einmal den Gehäusedeckel auf das System und kontrollieren Sie, ob die Laufwerke genau in den dafür vorgesehenen Ausschnitt im Gehäusedeckel passen.
Zentrieren Sie die Diskettenlaufwerke und schrauben Sie diese dann fest.
Die Flachbandkabel von den Laufwerken verbinden Sie mit der Laufwerkssteuerkarte (Controller), wobei das linke Laufwerk das Laufwerk 1 oder A und das rechte Laufwerk 2 oder B sein sollte. Eine entsprechende Beschriftung finden Sie an den Steckerleisten des Controllers. Wenn das Kabel von den Laufwerken zur Steuerkarte nicht richtig aufgesteckt wird, können an den Diskettenlaufwerken und am Controller erhebliche Schäden auftreten.

Wichtig: Achten Sie darauf, daß der Stecker richtig auf der Stiftleiste des Controllers sitzt. Das Kabel zeigt am Controller nach unten.

Setzen Sie nun die Steuerkarte in den Erweiterungssteckplatz 6 ein.
Die Flachbandkabel-Anschlüsse zeigen zur Rückwand.

Je nach eingesetztem Betriebssystem sind die üblichen Plätze für weitere Diskettenlaufwerke die Slots (Steckleisten) 4, 5 und/oder 7. Achten Sie hier bitte auf die Angaben in den entsprechenden Betriebshandbüchern.
Da die weiteren Laufwerke nicht eingebaut werden, müssen die Flachbandkabel durch den Durchbruch auf der Rückseite von den Laufwerken zu den Steckkarten geführt werden.

Pflege der Diskettenlaufwerke und der Disketten

Diskettenlaufwerke sind mechanische Geräte mit Motoren und anderen, sehr empfindlichen beweglichen Teilen. Daher sind sie etwas anfälliger als der BASIS 108 ohne Laufwerke. Rauhe Behandlung, wie Stöße, können zu Beschädigungen führen.

Die Diskette ist eine Plastikscheibe mit einer Beschichtung ähnlich der eines Tonbandes. Auf der Oberfläche können Informationen gespeichert oder von dort wieder abgerufen werden.
Die Diskette ist zum Schutz vor Staub und Kratzern in einer schwarzen Plastikhülle eingeschweißt. Innerhalb dieser Hülle kann sich die Diskette frei drehen.
Obwohl die Diskette relativ flexibel ist, vermeiden Sie bitte Verbiegen oder Knicke. Behandeln Sie auch die Hülle sorgfältig und stecken Sie sie sofort nach Gebrauch wieder in die zu jeder Diskette gehörende Papiertasche.

Vermeiden Sie jegliche Berührung der Oberfläche der Diskette.

Fassen Sie die Diskette nur an ihrer Hülle an.

Ein unsichtbarer Kratzer an der Oberfläche der Diskette oder lediglich ein Fingerabdruck können schon Fehler hervorrufen.

Legen Sie Disketten niemals auf schmutzige oder fettige Oberflächen und lassen Sie sie nicht verstauben.

Verwenden Sie einen Filzstift zum Beschriften der Diskettenaufkleber, wobei der Aufkleber erst nach dem Beschriften auf die Diskette geklebt werden sollte.

Halten Sie Disketten von Magnetfeldern fern, legen Sie sie nicht auf Bildschirmgeräte.

Disketten sind sehr empfindlich gegen extreme Temperaturen. Legen Sie sie nie in die Sonne oder in unmittelbare Nähe anderer Heizquellen, da sich die Disketten sonst wellen und nicht mehr gelesen werden können.
Bei sorgfältiger Pflege haben Disketten eine lange Lebensdauer.

Einlegen und Herausnehmen von Disketten

Das Laufwerk wird geöffnet und die Diskette mit dem Aufkleber nach oben hineingeschoben, wie es auf der Abbildung zu sehen ist. Die Kante mit dem ovalen Ausschnitt in der Hülle muß dabei zuerst hineingeschoben werden.
Schieben Sie die Diskette langsam hinein, bis sie vollständig im Laufwerk steckt. Biegen Sie sie dabei auf keinen Fall und schieben Sie nicht zu fest. Schließen Sie die Laufwerksklappe.



Einlegen der Diskette

Die Diskette wird herausgenommen, indem Sie das Laufwerk öffnen und die Diskette

vorsichtig herausziehen. Beim Öffnen der Laufwerksklappe wird auch gleichzeitig der Andruck für den Lese-/Schreibkopf gelöst. Es kann aber evtl. noch weiter geschrieben werden, was zu Datenverlust führen kann.

Wichtig: Nehmen Sie niemals eine Diskette aus dem Laufwerk, solange die rote Lampe des Laufwerks leuchtet, das kann die abgespeicherten Informationen zerstören.

Wenn Sie eine Diskette im Laufwerk lassen wollen, ohne mit dem System zu arbeiten, so empfiehlt es sich, die Laufwerksklappe zu öffnen, so daß der Kopf nicht auf der Diskette aufliegt.

KAPITEL 2

INHALTSVERZEICHNIS

Software

SOFTWARE

Einleitung

Das BASIS 108 System ist mit einem Monitor ROM ausgestattet, der das System automatisch startet, s. auch Kapitel 4. Damit haben Sie Zugriff zu den in diesem Kapitel beschriebenen Möglichkeiten des Monitor ROMs und können Ihre eigenen Betriebssysteme aufbauen.
Wahrscheinlich wird es allerdings so sein, daß Sie auf ein vorhandenes oder beim Kauf des BASIS 108 gleichzeitig erworbenes Betriebssystem zurückgreifen, um in einer der herkömmlichen Programmiersprachen auf Ihrem BASIS 108 arbeiten zu können.
Diese Betriebssysteme sind in der Regel auf Disketten abgelegt. Die Arbeit mit einem Kassettenrekorder ist möglich, aber sehr zeitaufwendig.
Die Betriebssysteme stellen im Prinzip nichts anderes dar als Arbeitshilfen, die es Ihnen ermöglichen auf einfachere und zugänglichere Weise mit Ihrem Computer zu sprechen. D.h. es hat Ihnen schon jemand die Arbeit des Umarbeitens Ihrer Programme in eine dem Computer verständliche Sprache abgenommen.
Grundsätzlich ist es so, daß diese Betriebssysteme in entsprechender Weise geladen werden müssen.
Hier sollen nicht alle möglichen Betriebssysteme angesprochen werden, sondern nur die nach unserer Erfahrung gebräuchlichsten:

UCSD p-System IV.0,
CP/M,
DOS3.3.

Es sei nochmals darauf hingewiesen, daß eine ganze Reihe anderer Betriebssysteme auf dem BASIS 108 möglich sind, eventuell ist eine vorherige Anpassung notwendig. Spezialfälle sollten Sie über Ihren Händler erfragen.
So sind alle Betriebssysteme, die für den Apple II angeboten werden oder die Sie von diesem Gerät noch besitzen, kompatibel. Das Apple Pascal ist das UCSD p-System II.1 und damit eine Teilmenge aus dem hier besprochenen UCSD p-System IV.0. Unterschiede im Betrieb werden kurz angesprochen. Die Firma Apple entwickelte speziell für die Umgebung von Basic das DOS-System, das nach Erstellen der ZAP-Diskette, s. Anhang A, vollständig kompatibel ist.
Auf die über diese Betriebssysteme möglichen Programmiersprachen können wir im Rahmen dieses Handbuches nicht eingehen, die gängigsten Sprachen in den einzelnen Systemen werden aber entsprechend erwähnt.
Andererseits soll dieses Kapitel nicht die Betriebshandbücher ersetzen, sondern Ihnen die Möglichkeiten aufzeigen und Ihnen eventuell die Entscheidung für das eine oder andere System erleichtern.

UCSD p-System IV.0

Das UCSD IV.0 Betriebssystem ist ein Programmentwicklungswerkzeug für Microcomputersysteme, erstellt von der University of California San Diego.

Für den BASIS 108 steht Ihnen eine Interpreter Implementation des UCSD IV.0 Pascal zur Verfügung. Das bedeutet, daß ein Compiler Ihre Programme in einen Pseudo-Code (P-Code) übersetzt. Dieser Code ist unabhängig vom jeweiligen Mikroprozessor. Während der Ausführung des Programmes wird der P-Code durch ein Assemblerprogramm interpretiert und auf dem 6502 Prozessor des BASIS 108 ausgeführt. Auch die Module des Betriebssystems sind Pascalprogramme und werden in der gleichen Weise wie die Benutzerprogramme ausgeführt.

Es besteht aus den Programm-Modulen Editor, Compiler, Linker, Assembler, Filer und einem Debugger.
Wenn Sie Ihr System starten, erscheinen in der oberen Bildschirmzeile die System Kommandos, mit denen Sie durch Drücken des Anfangsbuchstaben die obigen Programm-Module anwählen können.
Kommando-Zeile:

Command: E(dit,R(un,C(omp,L(ink,X(ecute,A(ssem, D(ebug,? [IV.0 B3n]

Beschreibung der Kommandos:

E

ruft den bildschirmorientierten Texteditor auf, der eine recht komfortable Textverarbeitung zuläßt. Der bearbeitete Text wird vom Betriebssystem nach Abschluß der Textbearbeitung unter dem Namen SYSTEM.WRK.TEXT auf der Diskette gesichert und wird im folgenden mit Workfile bezeichnet.

R

übersetzt den Workfile, sofern es ein Programm in einer höheren Sprache ist, durch den Compiler in den P-Code und führt das Programm anschließend aus. Entspricht der Text nicht der Syntax, so erfolgt eine Fehlermeldung. Ist die Übersetzung des Workfile in den P-Code erfolgreich, so wird dieser Codefile unter dem Namen SYSTEM.WRK.CODE abgespeichert. Dieser Codefile kann jederzeit über R ausgeführt werden.

F

Startet das Programm-Modul Filer und es erscheint eine neue Kommandozeile:

Filer: G(et,S(ave,W(hat,N(ew,L(dir,R(em,C(hng,T(rans,D(ate,? [C.12a]

Mit den Filerkommandos verwalten Sie das aktuelle Datum, ihren Arbeitsfile (sichern, löschen, bestehende Files bearbeiten) und ihre Programme. Sie können Programme transferieren, Programmnamen ändern und sich den Inhalt der Disketten ansehen (näheres siehe Betriebshandbuch).

C

Startet das Program-Modul Compiler, das den anzugebenden Programmtext xxx.TEXT einer höheren Programmiersprache in den P-Code übersetzt und bei erfolgreicher Compilierung unter xxx.CODE sichert. xxx ist der Name den der Benutzer selbst festlegt.

L

Ruft das Programm-Modul Linker auf, welches den P-Code mit dem echten Maschinen-Code verbindet. Es wird vornehmlich zum Verbinden von Assemblerroutinen mit Hauptprogrammen höherer Programmiersprachen benötigt.

X

Durch dieses Kommando werden übersetzte Programme, die unter dem Namen xxx.CODE auf der Diskette verfügbar sind, ausgeführt.

A

Assemblerprogramme, die mit dem Texteditor erstellt worden sind, werden in einen echten Maschinen-Code übersetzt und können mit dem Linker in Hauptprogramme höherer Programmiersprachen eingebunden werden.

D

Der Debugger ist eine zusätzliche Hilfe bei der Fehlersuche in bereits compilierten Programmen. Er kann von der Kommandozeile aus und auch während der Programmausführung aufgerufen werden und erleichtert das Auffinden von Fehlern, die der Compiler nicht berücksichtigt (z.B. logische Fehler im Programmablauf).

Die Leistungsfähigkeit des Betriebssystems UCSD IV.0 wird durch die Verfügbarkeit von Bibliotheksprogrammen unterstrichen.
Proceduren und Functions, die häufig benötigt werden, können in der System-Bibliothek abgelegt werden (SYSTEM.LIBRARY). Programme höherer Programmiersprachen können nun diese Routinen benutzen.

Inhaltsverzeichnis der vier notwendigen Disketten

```
108.1:
SYSTEM.BOOT          10   31-May-82
SYSTEM.SBIOS          7   31-May-82
SYSTEM.INTERP        28   28-May-82
SYSTEM.MISCINFO       1   27-May-82
SYSTEM.FILER         33   19-Oct-81
SYSTEM.LIBRARY       11   28-Jan-82
SYSTEM.SYNTAX        14    4-Dec-80
SYSTEM.PASCAL       103    3-Jun-82
SYSTEM.WRK.TEXT       4    3-Jun-82
SYSTEM.WRK.CODE       2    3-Jun-82
```

```
108.2:
SYSTEM.COMPILER      96    5-Jan-82
SYSTEM.SYNTAX        14    4-Dec-80
SYSTEM.EDITOR        49    7-Dec-81
LIBRARY.CODE         13    7-Dec-81
SUNITS.LIBRARY       52   31-May-82
ID.TEXT               4   31-May-82
KEYWORDS.TEXT         4   31-May-82
WINDOW.CODE           2   25-May-82
DISPLAY.CODE          2   25-May-82
WINDOW.TEXT           4   25-May-82
DISPLAY.TEXT          4   25-May-82
```

```
108.3:
SYSTEM.ASSEMBLER     46    7-Dec-81
6500.OPCODES          2   20-Dec-78
6500.ERRORS           7   23-Sep-80
SYSTEM.LINKER        26    7-Dec-81
SYSTEM.EDITOR        49    7-Dec-81
LIBRARY.CODE         13    7-Dec-81
COMPRESS.CODE        10    7-Dec-81
```

```
108.4:
SETUP.CODE           27    7-Dec-81
BOOTER.CODE           3    7-Dec-81
DISKCHANGE.CODE       8    7-Dec-81
DISKSIZE.CODE         3    7-Dec-81
FINPARAMS.CODE        9    7-Dec-81
ABSWRITE.CODE         4    7-Dec-81
YALOE.CODE           12    7-Dec-81
SCREENTEST.CODE      13    7-Dec-81
DECODE.CODE          28    5-Mar-81
COPYDUPDIR.CODE       3    7-Dec-81
MARKDUPDIR.CODE       4    7-Dec-81
PATCH.CODE           34    7-Dec-81
COMPRESS.CODE        10    7-Dec-81
XREF.CODE            28    7-Dec-81
RECOVER.G.CODE        8    7-Dec-81
FORMATTER.CODE       14    7-Jan-82
```

Dies ist der Stand vom 18.6.1982. Sollten Sie neuere Versionen besitzen, so sind Abweichungen im Interesse des Fortschrittes möglich.

Das CP/M-System

CP/M (Control Program for Microprocessors) der Firma Digital Research, USA, ist ein Steuerprogramm für Mikrocomputersysteme mit Disketten- und/oder Festplattenlaufwerken, speziell für Computer, die einen 8080/8085 oder Z-80 als Zentraleinheit haben und über mindestens 16 KByte Hauptspeicher verfügen. Beides trifft für den BASIS 108 zu.
Während Sie bei den UCSD-Systemen über das Drücken der jeweiligen Buchstabentaste den Befehlsablauf steuern, rufen Sie beim CP/M-System die jeweilige gewählte Funktion über das zusätzliche (RETURN) ab.
Die spezielle Sammlung von CP/M-Programmen machen durch einfache Systembefehle dem Benutzer alle vom Computer gesteuerten Hardwarekomponenten zugänglich. CP/M verwaltet darüber hinaus alle internen und externen Einheiten, unter anderen auch alle verfügbaren Speicherkapazitäten der Disketten und des Arbeitsspeichers, vollkommen selbständig.
In den Arbeitsspeicher des Systems geladen, bildet CP/M einen integrierten Bestandteil des gesamten Systems. Der Benutzer kann mit CP/M in Dialog treten und beliebige Anwendungsprogramme starten.
CP/M ist in drei Funktionsmodule aufgeteilt:

CCP (Console Command Prozessor),
BDOS (Basic Disk Operating System),
BIOS (Basic Input/Output System).

CCP liest die Tastaturkommandos und erzeugt BDOS-System-Aufrufe.
Zum Lesen und Arbeiten von Programmiersprachen benötigt CP/M wie auch das oben besprochene UCSD p-System IV.0 einen entsprechenden Compiler oder Interpreter. Damit ist es dann möglich, praktisch in allen gängigen Programmiersprachen zu arbeiten, wobei das CP/M-System die Organisation übernimmt.
Ferner besitzt CP/M die Möglichkeit zum Assemblieren von Programmen und zum Einordnen von Asemblerprogrammen in die jeweils laufenden Programme.
Die Zahl der möglichen höheren Programmiersprachen ist sehr groß. Es gibt ausgezeichnete Textsysteme und andere Anwenderprogramme, so daß man hier ebenfalls ein umfassendes Betriebssystem zur Verfügung hat.

Im folgenden werden einige häufig vorkommende Kommandos aufgeführt und kurz beschrieben:

ASM Assemblieren (8080) einer Datei.

DDT Testen und Ändern von 8080-Maschinenprogrammen.

DIR Anzeigen einer Liste aller auf der Diskette des selektierten Laufwerks verzeichneten Dateien.

ERA Löschen einer oder mehrerer Dateien auf der Diskette.

PIP Kopieroperationen von Dateien.

SAVE Sichern eines Speicherinhaltes als Disk-Datei.

REN Umbenennen einer Datei.

SUBMIT Ausführen einer Befehlsfolge.

Die Anwendung dieser und weiterer Programme entnehmen Sie bitte einem CP/M-Betriebshandbuch.

Es folgt der Inhalt der Diskette, die das CP/M-Betriebssystem enthält:

```
A: FORMAT   COM : DEUTSCH   COM
A: ASCII    COM : APL       COM
A: SYSWRT   BAS : PIP       COM
A: STAT     COM : ED        COM
A: ASM      COM : DDT       COM
A: LOAD     COM : SUBMIT    COM
A: XSUB     COM : DUMP      ASM
A: XSUB     COM
```

Auch hier können sich Änderungen ergeben, Version vom 18.6.1982.

Das DOS3.3-System

Um im DOS3.3 arbeiten zu können, muß es zunächst auf den BASIS 108 angepaßt werden. Das geschieht entsprechend Anhang A einmal. Dann geben Sie zunächst die ZAP-Diskette in Ihr Laufwerk 1, wählen die entsprechende Basic-Art und können dann nach Eingabe Ihrer DOS-Diskette arbeiten wie z.B. auf einem Apple, wenn Sie einige kleine Änderungen berücksichtigen.
Wie schon erwähnt, handelt es sich beim DOS3.3 eigentlich nicht um ein echtes Betriebssytem, sondern eher um eine Umgebung für Basic. D.h., hiermit lassen sich praktisch nur die entsprechenden Basic-Arten bearbeiten. Andererseits haben Sie hier die Möglichkeit, über entsprechende Befehle das Monitor ROM anzusteuern und in ihm zu arbeiten, s. Kapitel 4.
Da es aber eine Vielzahl von Anwenderprogrammen in Basic gibt, die speziell auf das DOS-System ausgelegt sind, ist auch dieses System attraktiv.

Die häufigsten Befehle mit einer kurzen Beschreibung:

BRUN X Lädt Maschinen-Programm X in Speicher und läßt es ablaufen.

CATALOG Gibt den Inhalt der im aktuellen Laufwerk liegenden Diskette an.

DELETE X Entfernt Programm X von der Diskette.

IN # n Steuert Slot n für Eingabe an.

LOAD X Lädt Basic-Programm X in den Speicher.

PR # n Steuert Slot n für Ausgabe an.

RUN X Lädt Basic-Programm X in Speicher und läßt es ablaufen.

SAVE X Speichert Basic-Programm X auf Diskette.

In dem entsprechenden Betriebshandbuch für DOS finden Sie diese und weitere Befehle und Funktionen und Ihre Anwendung. Hier ist also wie bei den anderen beiden Betriebssystemen nur ein kleiner Ausschnitt aus den Möglichkeiten aufgeführt.

Die hier im folgenden abgedruckte Inhaltsliste der DOS3.3 SYSTEM MASTER Diskette enthält nicht die möglichen Spiele oder Demonstrationsprogramme:

```
A  006  DOS3.3
B  010  BOOT13
I  009  COPY
B  003  COPYA.OBJ0
A  009  COPYA
B  020  FID
B  050  FPBASIC
B  050  INTBASIC
B  009  MASTER CREATE
B  027  MUFFIN
A  010  RANDOM
A  013  RENUMBER
A  002  DISPLAY
B  002  DISPLAY SPEC
A  006  BAUD
A  006  PRINTER/V24
```

Auch hier können sich je nach der verwendeten Version Änderungen ergeben.

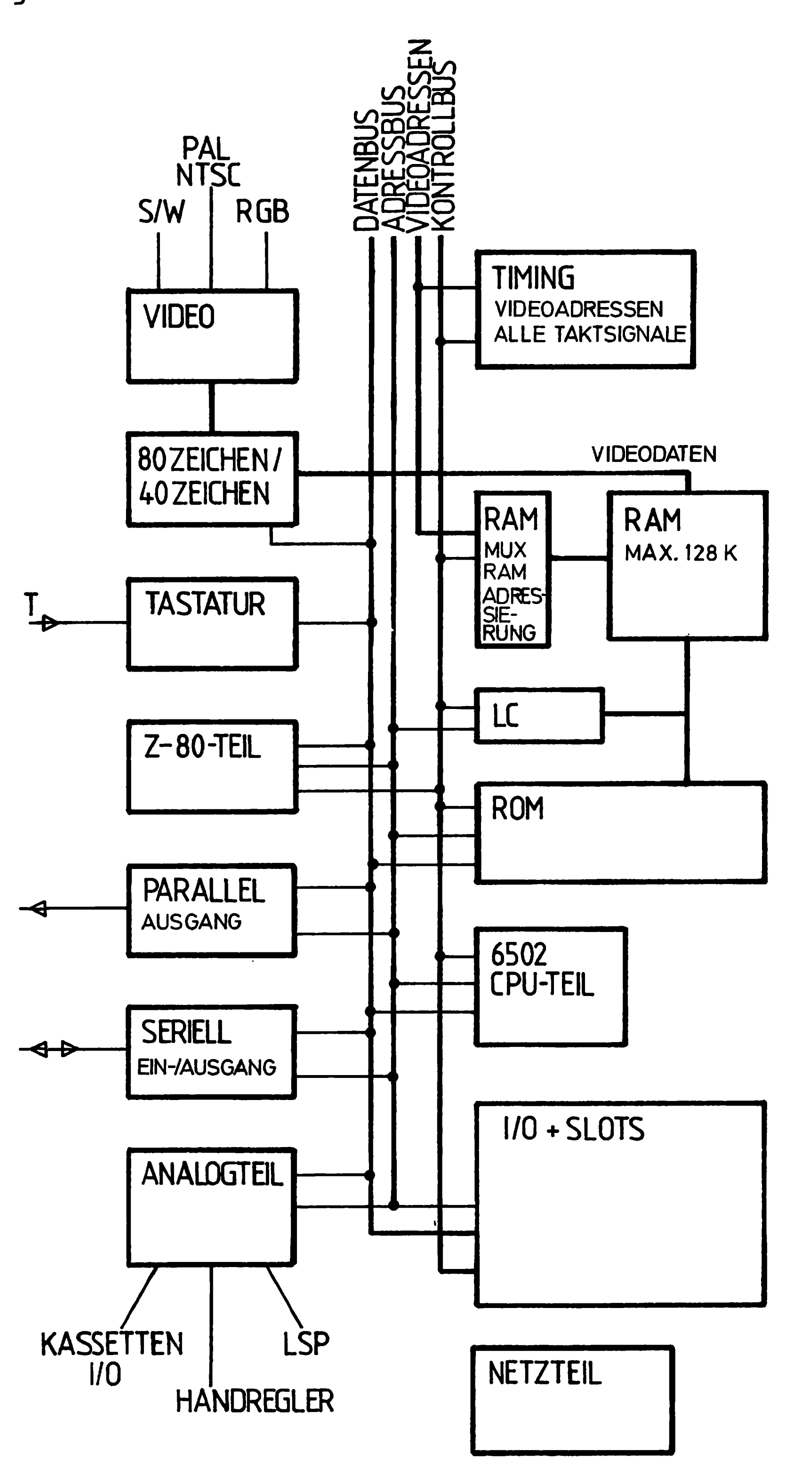
KAPITEL 3

INHALTSVERZEICHNIS

Zugriff zur Hardware

Logischer Schaltplan

Zum besseren Verständnis der folgenden Kapitel wird hier zunächst der logische Schaltplan aufgeführt.



Text- und Graphikdarstellung

Das BASIS 108 Computersystem kann sowohl Text als auch Graphik darstellen. Zur Darstellung von Text oder LO-RES-Graphik (niedrige Auflösung) und MI-RES-Graphik (mittlere Auflösung) stehen 2 Bereiche (Seiten) und für die HI-RES-Graphik (hohe Auflösung) zwei weitere Bereiche zur Verfügung. Diese Bereiche sind direkt im Adressraum der Microprozessoren untergebracht.

Der Textbildschirm kann entweder 40 oder 80 Zeichen in 24 Zeilen - je nach ausgewähltem Mode - darstellen. Die gleichen Seiten werden auch für die niedrig auflösende Graphik genutzt, so daß sich im Graphik Mode entweder 40 x 48 Blöcke oder 80 x 48 Blöcke in 16 Farben darstellen lassen.

Ein weiterer Bereich des Speichers wird für 2 Seiten der HI-RES-Graphik mit einer Auflösung von 280 x 192 Punkten in 6 Farben genutzt.

1. Textseite $0400-$07FF (Text oder LO-, MI-RES-Graphik)
2. Textseite $0800-$BFFF

1. Graphikseite $2000-$3FFF (HI-RES-Graphik).
2. Graphikseite $4000-$5FFF

Der Textbildschirm

Die erste Seite des Textbildschirmes liegt auf der Adresse $0400 und reicht bis zur Adresse $07FF, die zweite Seite schließt direkt mit der Adresse $0800 an und reicht bis zu Adresse $OBFF. Über die Softwareschalter $C054 (Seite 1) und $C055 (Seite 2) kann die jeweils auf dem Bildschirm darzustellende Seite ausgewählt werden.

80/40-Zeichendarstellung

Für die 80-Zeichendarstellung wurde dem Adressbereich der beiden Textseiten ein 2 KByte statisches RAM parallel geschaltet. Dieses statische RAM wird mit den gleichen Adressen angesprochen wie auch die normalen Textseiten. Beim Schreiben in die Textseiten wird über einen Softwareschalter die entsprechende Seite ausgewählt.

$C00Dw aktiviert das statische RAM,
$C00Cw aktiviert den Standard-Bereich.

Durch diesen Schalter werden immer beide Textseiten umgeschaltet.

Die Adresse $C00Bw schaltet die 80-Zeichen Darstellung ein und $C00Aw wieder aus. Das statische RAM kann aber unabhängig von diesem Schalter beschrieben oder gelesen werden.

Das Prinzip der 80-Zeichendarstellung

Der Bildschirmwiederholungsspeicher kann nur in den Augenblicken ausgelesen werden, in denen der Mikroprozessor keine Speicherzugriffe durchführt. Dieses ist immer der Fall, wenn der Takt des Prozessors auf logisch 0 liegt. Dieses wird nun genutzt, um ein Zeichen für den Bildschirm aus dem Speicher zu lesen. Die Darstellung von 80 Zeichen in einer Zeile würde aber verlangen, daß auch während der anderen Taktphasen ein Zeichen gelesen werden muß. Damit die Kompatibilität zum Apple erhalten bleibt, ist dies aber ohne wesentliche Veränderung nicht möglich. Im BASIS 108 werden deshalb 2 Zeichen gleichzeitig ausgelesen. Ein Zeichen aus dem Standard RAM und ein Zeichen aus dem statischen RAM. Diese Zeichen werden zwischengespeichert und können dann unabhängig vom Mikroprozessortakt weiter verarbeitet werden.
Diese Technik bedingt, daß sich alle Zeichen mit einer geraden Platznummer im Standard RAM und alle mit einer ungeraden im statischen RAM befinden. Das statische RAM kann, wenn es selektiert wurde, vom Mikroprozessor ausgelesen werden.

Softwareschalter für die Textdarstellung

$C054	Seite 1 aktiv,
$C055	Seite 2 aktiv,
$C00Aw	80 Zeichendarstellung aus,
$C00Bw	80 Zeichendarstellung ein,
$C00Dw	statisches RAM selektiert,
$C00Cw	Standard RAM selektiert

Softwareschalter für die Graphik

LO-RES-Graphik

Die LO-RES-Graphik benutzt die gleichen Bereichen, wie die Textseiten und ist daher ebenfalls auf 2 Seiten vorhanden. In dieser Graphikart können entweder 40 x 48 Blöcke in 16 Farben (Vollgraphik) oder 40 x 40 Blöcke mit 4 Zeilen Text am unteren Bildschirmrand (mixed Graphik) dargestellt werden. Die Anwahl geschieht mit Hilfe von Softwareschaltern.

Schalter für die LO-RES-Graphik:

$C050	Graphik einschalten,
$C051	Graphik ausschalten,
$C056	LO- + MI-RES-Graphik,
$C053	mixed (4 Zeilen Text werden eingeblendet),
$C052	Vollgraphik (die Textzeilen werden ausgeblendet),
$C00Aw	80 Spalten aus.

MI-RES-Graphik

Die MI-RES-Graphik stellt 80 x 48 Blöcke oder 80 x 40 Blöcke in 6 Farben dar. Sie besitzt die selben Möglichkeiten, wie die LO-RES-Graphik, nur wird das statische RAM zur 80 Zeichendarstellung mitverwendet. Es gelten die gleichen Bedingungen für das statische RAM wie bei der 80 Zeichen Textdarstellung.

Schalter für die MI-RES-Graphik:

$C050 Graphik einschalten,
$C051 Graphik ausschalten (nur Text),
$C056 LO- + MI-RES-Graphik ein, HI-RES aus,
$C053 mixed (4 Zeilen Text werden eingeblendet),
$C052 Vollgraphik,
$C00Bw 80 Spalten ein.

Weiterhin sind für die Programmierung noch die Schalter $C00D und $C00C für das Beschreiben des statischen RAMs notwendig.

HI-RES-Graphik

Die HI-RES-Graphik ist eine hochauflösende Farbgraphik mit 280 x 192 Punkten in 6 Farben. Auch diese Graphikart hat 2 Seiten im Speicher; Seite 1 im Adressbereich $2000 bis $3FFF und Seite 2 von $4000 bis $5FFF. Die HI-RES-Graphik kann als Vollgraphik (280 x 192 Punkte) oder als mixed Graphik (280 x 160 Punkte) mit 4 Zeilen Text am unteren Bildrand betrieben werden. In diesem Mode wird als Text der Inhalt der entsprechenden Textseite mit 40 oder 80 Zeichen pro Zeile eingeblendet.

Schalter für die HI-RES-Graphik:

$C050 Graphik einschalten,
$C051 Graphik ausschalten (Text ein),
$C057 HI-RES-Graphik ein,
$C053 mixed HI-RES-Graphik,
$C052 Vollgraphik.

Farbdarstellung der HI-RES-Graphik

Jeder Punkt auf dem Bildschirm repräsentiert ein Bit aus dem Bildspeicher. Von den 8 Bit eines jeden Bytes werden die Bits 0 . . . 6 auf dem Bildschirm dargestellt, das Bit 7 bestimmt die Farben der Punkte in diesem Byte. Auf einem S/W Bildschirm erscheint ein Punkt, wenn das Bit logisch 1 ist, und kein Punkt, wenn es logisch 0 ist.
Auf einem Farbbildschirm ist dies nicht ganz so einfach. Hier ist die Bit-Position für die dargestellte Farbe wichtig. Ist ein Bit auf einer ungeraden Position an, stellt es entweder grün oder hellblau dar. Ist ein Bit auf einer geraden Position an, ergeben sich die Mischfarben aus rot und grün oder aus hellblau und violett. Die zweite Kombination (hellblau, violett) ist nur dann gültig, wenn das 8 Bit des

entsprechenden Bytes an ist. Innerhalb eines Bytes ist es nicht möglich, die Farbgruppe zu wechseln. Die hier genannten Farben können je nach Bildschirmtyp und Einstellung voneinander abweichen.

Zeichengenerator

Im BASIS 108 Computersystem ist der Zeichengenerator in einem 4 KByte EPROM (2732 Typ) untergebracht. In diesem EPROM können bis zu 5 Zeichensätze untergebracht werden. Durch 4 Softwareschalter kann der gewünschte Zeichensatz ausgewählt werden. Wenn der Schalter SW 3 ($C006) auf logisch 1 steht, ist ein Zeichensatz mit 128 Zeichen normal, 64 Zeichen invertiert und 64 Zeichen blinkend ausgewählt. Ist dieser Schalter auf logisch 0, können 4 weitere Zeichensätze angewählt werden.

Zeichengenerator

			SW0	SW1	SW2	SW3
Satz 0	Standard Apple II	64 Zeichen	0	0	0	0
Satz 1	Standart ASCII	128 Zeichen	x	1	0	0
Satz 2	Deutsch	128 Zeichen	x	0	1	0
Satz 3	APL	128 Zeichen	x	1	1	0

(In Ländern außerhalb des deutschen Sprachraumes kann Satz 2 und 3 vertauscht sein.)

Adresse	Schalter	
$C000w	SW0	aus
$C001w	SW0	ein
$C002w	SW1	aus
$C003w	SW1	ein
$C004w	SW2	aus
$C005w	SW2	ein
$C006w	SW3	aus
$C007w	SW3	ein,

EIN entspricht logisch 1 und AUS logisch 0.

Der Schalter SW0 kann den Zeichensätzen 1-3 entweder INVERSE oder FLASHING (Blinken) als Sonderdarstellung zuordnen:

$C000w Inverse,
$C001w Flash.

Die Tastatur

Die Tastatur besteht aus einer erweiterten Schreibmaschinentastur, einem numerischen Zehnerblock mit Tasten für die Grundrechenoperationen, einem Cursorsteuerfeld und 15 Zusatztasten. Sie ist in einem sehr flachen Kunststoffgehäuse untergebracht und mit dem BASIS 108 über ein 16-adriges Kabel verbunden. Die Steckerbelegung des Tastaturkabels finden Sie auf Seite 8.
Die 15 Zusatztasten sind vierfach belegt. Sie werden wie normale Tasten verwandt, gehen aber über den ASCII-Zeichensatz hinaus. Unter CP/M 3.0 können sie softwaremäßig mit einigen Funktionen belegt werden.
Groß-/Kleinschreibung wird durch die SHIFT-Taste erreicht, die durch Drücken der Taste LOCK festgesetzt wird, SHIFT-LOCK. Im alphanumerischen Tastenfeld sind Umlaute und Sonderzeichen vorhanden. Sollen nur die Buchstaben großgeschrieben werden, die übrigen Tasten aber mit ihrer unteren Belegung erscheinen, so drückt man gleichzeitig CTRL-LOCK. Alle Tasten sind mit Autowiederholung ausgerüstet, das bedeutet, daß sich bei längerem Druck auf die Taste das gedrückte Zeichen automatisch wiederholt.

BASIS 108 Tastatur

Anzahl der Tasten	100
Codierung	ASCII mit Sonderzeichen
Anzahl der Tastencodes	128 ASCII und 70 Funktionen
Ausgang	TTL
Betriebsspannung	+12 Volt

Die Dekodierung der Tastenbelegung erfolgt auf der Hauptplatine des BASIS 108 in einem ROM. Hierdurch ist eine flexible Tastenbelegung durch Austauschen des ROMs gegeben. Außerdem besteht die Möglichkeit, über einen Softwareschalter ($C009 ein, $C008 aus) die Abfrage der Tastatur über einen Interrupt zu steuern.

Auf einer zweiten Eingabeadresse kann der geübte Programmierer den Status bestimmter Funktionen der Tastatur abfragen (siehe auch nächste Seite):

- CONTROL-Taste gedrückt,
- SHIFT-Taste gedrückt,
- Funktionstaste gedrückt.

Außerdem können auf dieser Adresse noch die folgenden Statusinformationen, die nicht mit der Tastatur zusammenhängen, abgefragt werden:

- Printer Return,
- HBL (Horizontal Austastsignal),
- Sync (Video Synchronisationssignal).

Die RESET-Funktion wird durch das gleichzeitige Drücken der beiden SHIFT- und der CRTL-Taste ausgelöst. Der Programmablauf wird unterbrochen und beim Loslassen der Tasten startet der Computer den RESET-Ablauf.

Adressen der Tastatur

$C000-$C007	Lesen des ASCII-Code der Taste
$C008-$C00F	Lesen des Status der Tastatur
$C009w	Tastaturinterrupt ein
$C008w	Tastaturinterrupt aus

Das auf einer der Adressen $C008-$C00F gelesene Byte besitzt folgende Zuordnung:

Bit 7	Zusatztaste
Bit 6	Shifttaste
Bit 5	Controltaste
Bit 4	z.Z. nicht definiert
Bit 3	z.Z. nicht definiert
Bit 2	HBL (Horizontal Austastsignal)
Bit 1	Video Synchronisationssignal
Bit 0	Drucker aktiv

Im Anhang I finden Sie die Belegung der Tasten mit den einzelnen Zeichen und eine Tabelle in der die Tasten, der Code (hexadezimal) und das ASCII Zeichen aufgelistet sind.

KAPITEL 4

INHALTSVERZEICHNIS

Der Monitor

Einleitung

Der System-Monitor, ein kleines aber leistungsstarkes Programm, befindet sich auf der Hauptplatine in einem ROM (Read Only Memory). Das Monitor-ROM hat eine Speicherkapazität von 2 KByte und sitzt auf dem IC-Platz 25, in der vorderen Reihe auf der Platine. Mit Hilfe dieses Programmes werden Abläufe im System kontrolliert und gesteuert.
Sie hatten beim Kauf Ihres BASIS 108 die Wahl zwischen zwei unterschiedlichen Monitoren. Das eine Monitor-ROM ist für den Einsatz des BASIS 108 mit Laufwerk vorgesehen und der andere Monitor-ROM beinhaltet die Schreib-/Leseroutinen für einen Kassettenrekorder. Eine Tabelle mit den Unterschieden der beiden Monitor-ROMs finden Sie im Anhang N. Beschreibung des Monitor-ROM für Kassettenrekorder (40 Zeichen/Zeile) mit den entsprechenden Routinen finden Sie im Anhang G.
Das Monitor-ROM erfüllt in den verschiedenen, möglichen Betriebssystemen unterschiedliche Aufgaben. Während es für das Betriebssystem UCSD IV.0 bzw. Apple Pascal nur für den Ladevorgang benötigt wird, wird es von den beiden weiteren üblichen Systemen CP/M und DOS ständig benötigt. Siehe auch die entsprechenden Betriebshandbücher.

Einweisung

Das Programm des BASIS 108 Monitor-ROMs beginnt ab der Adresse $FF69 (dezimal 65385 oder -151).
Aus einem BASIC-Programm können Sie den Monitor-ROM mit dem Befehl CALL -151 aufrufen. Haben Sie FP40 oder FP80 geladen, so kann das Monitor-ROM auch über SYS angesprochen werden.
Aus dem Betriebssystem CP/M kann man das Monitor-ROM nicht so einfach ansprechen. Bitte lesen Sie für diesen Fall das entsprechende CP/M-Handbuch.
Haben Sie kein System geladen so können Sie durch Abstellen des Laufwerkkontrollers mit (RESET) ebenfalls in den Monitor-ROM gelangen.
Der Monitor meldet sich auf dem Bildschirm mit einem Stern * und rechts daneben der Cursor. Damit wird angezeigt, daß das Monitor-Programm bereit ist, von Ihnen einen Befehl zu empfangen.

Ihre Eingaben über die Tastatur dürfen bis zu 255 Zeichen lang sein und müssen mit (RETURN) beendet werden. Wenn Sie den Monitor verlassen wollen und zu der Sprache zurückkehren wollen, mit der Sie eben gearbeitet haben, dann drücken Sie Q oder System-(RESET) (gleichzeitig SHIFT-SHIFT-CTRL).

Daten und Adressen

Die Kommunikation mit dem Monitor erfolgt über die Tastatur oder Ihr Programm. Sie tippen eine Zeile auf der Tastatur und geben danach (RETURN) ein. Der Monitor wird das verarbeiten, was Sie ihm eingegeben haben. Er kann folgende Arten an Informationen verwerten: Adressen, Werte und Befehle (Kommandos).

Adressen und Werte nimmt er nur in hexadezimaler Schreibweise an. Diese hexadezimale Schreibweise wird im Anhang H näher dargestellt.
Jede Adresse im BASIS 108 wird durch vier Hexadezimalziffern dargestellt und jeder Wert, Inhalt einer Speicherstelle, durch zwei Hexadezimalziffern. Wenn der Monitor die Eingabe einer Adresse erwartet (Stern mit danebenstehendem Cursor), akzeptiert er jede Gruppe von Hexadezimalziffern. Sind weniger als vier Ziffern in dieser Gruppe, so wird er führende Nullen ergänzen, gibt es mehr als vier Ziffern, so werden nur die letzten vier Ziffern ausgewertet. Entsprechend behandelt der Monitor die Eingabe der zweiziffrigen Datenwerte.
Der Monitor erkennt 22 verschiedene Kommandos. Einige sind Satzzeichen, andere sind Buchstaben oder Steuerzeichen. Das Monitor-ROM benötigt, wie Sie es von den verschiedenen Betriebsystemen her kennen, nur den ersten Buchstaben eines Kommandos, ein Kommando wird durch Steuerzeichen aufgerufen.
! Obwohl der Monitor das Steuerzeichen CTRL-B erkennt und richtig
! interpretiert, wird es nicht auf dem Bildschirm sichtbar gemacht.

Inhaltsüberprüfung einer Speicherstelle

In den folgenden Abschnitten werden die von Ihnen einzugebenden Werte fett gedruckt, wobei die Antworten, die der Monitor auf dem Bildschirm darstellt, normal gedruckt, aber groß geschrieben sind.
Wenn Sie die Adresse einer Speicherstelle eingeben, wird der Monitor wie folgt antworten:

- Wiederholung der eingegebene Adresse,
- ein Doppelpunkt,
- ein Leerzeichen,
- den Wert dieser Speicherstelle.

Beim Überprüfen der folgenden Beispiele können die auf dem Bildschirm ausgegebenen Speicherinhalte, solange Sie sie nicht in der vorgeschriebenen Form geändert haben, von den hier gedruckten Speicherinhalten abweichen.

Beispiel:

```
*20(RETURN)
 0020: 00
*
```

Überprüfen mehrerer Speicherstellen

Wenn Sie dem Monitor in einer Eingabezeile einen Punkt . und darauffolgend eine Adresse angeben, erhalten Sie einen Speicherauszug von der zuletzt angezeigten Adresse bis zu der eingegebenen Adresse. Die letzte der dabei angezeigten Adressen ist die Startadresse für weitere Anzeigen.

Beispiel:

```
*0(RETURN)
 0000: 04
*.11(RETURN)
 0001: C6 00 0A 1B 18 18 00 00 FF 4C FF FF 22 00 6B
 0010: 00 00
*
```

Nachfolgend noch einige Bemerkungen zum Format eines Speicherauszugs:

1. Der Speicherauszug beginnt mit der auf die zuletzt angezeigten bzw. geöffneten Speicheradresse, im Beispiel oben also mit 0001.

2. Die anderen Zeilen beginnen mit Adressen, die mit einer Null enden.
   Bei dem Monitor-ROM, der mit 40 Zeichen/Zeile startet, sind die Zeilen aufgeteilt und beginnen mit Adressen, die mit einer 8 oder mit einer 0 enden.
3. Es werden entsprechend dem Monitor-ROM 8 bzw. 16 Werte in einer Zeile angezeigt.

Sie können die zwei Kommandos auf einmal eingeben: Tippen Sie die Anfangsadresse, dann einen Punkt und die Endadresse. Diese beiden Adressen, die durch einen Punkt getrennt wurden, nennt man Speicherbereich.

Beispiel:

```
*30.40(RETURN)
 0030: FF 00 FF AA 05 00 BD 9E 81 9E FF FF 36 00 41 00
 0040: 30 00
*
```

Ein Druck auf (RETURN) veranlaßt den Monitor, eine Zeile mit dem Speicherauszug anzuzeigen. Der Speicherauszug beginnt bei der Adresse, die der zuletzt angezeigten oder geöffneten Adresse folgt, und endet bei der Adresse, die mit einem F endet. Wieder wird die zuletzt angezeigte Adresse als die zuletzt geöffnete und als nächste veränderbare Adresse betrachtet.

Beispiel:

```
*5(RETURN)
 0005: 18
*(RETURN)
 18 00 00 FF 4C FF FF 22 00 6B
*(RETURN)
 0010: 00 00 00 00 04 00 FF 00 FF FF FF FF FF FF FF FF
*
```

Änderung einer Speicherstelle

In dem letzten Abschnitt haben Sie einiges über die nächste veränderbare Speicherstelle erfahren. Wenn Sie das folgende Beispiel durchführen, können Sie sehen, was wirklich passiert.
Tippen Sie einen Doppelpunkt und dann einen Wert.

Beispiel:

```
*0(RETURN)
 0000: 04
*:3C(RETURN)
*0(RETURN)
 0000: 3C
*
```

Sie sehen, daß der Wert des Speichers 0 den neuen Wert 3C hat.

Sie können das Öffnen und Ändern zu einer Anweisung zusammenfassen:

Beispiel:

```
*10:33(RETURN)
*10(RETURN)
 0010: 33
*
```

Wenn Sie den Inhalt einer Speicherstelle verändern, verliert sie den alten Wert. Der neue Wert bleibt solange erhalten, bis er wiederum von einem anderen Wert überschrieben wird.

Änderung von aufeinanderfolgenden Speicherstellen

Wenn Sie mehrere aufeinanderfolgende Speicherstellen verändern wollen, brauchen Sie nicht jede einzelne Speicherstelle so einzutippen, wie es im vorigen Abschnitt beschrieben wurde. Der Monitor ermöglicht es Ihnen, maximal 58 Speicherstellen auf einmal zu ändern. Dazu geben Sie die Anfangsadresse, einen Doppelpunkt und dann alle Werte ein.

Die Werte müssen durch Leerstellen voneinander getrennt sein.
Der Monitor trägt nun ab der angegebenen Anfangsadresse alle Werte der Reihe nach in die Speicherstellen ein. Wollen Sie noch mehr Speicherstellen ändern, brauchen Sie nicht die Adresse neu eingeben, sondern nur einen Doppelpunkt und die neuen Werte, sofern die Startadresse mit der nächsten, auf die zuletzt geänderten Adresse übereinstimmt.

Beispiel:

```
*0.7(RETURN)
 0000: 5F C6 00 0A 1B 18 18 00
*0: 6F 3A 1 B 1A 16 11 07 (RETURN)
*0.7(RETURN)
 0000: 6F 3A 01 0B 1A 16 11 07
*
```

Übertragen eines Speicherbereichs

Der Inhalt eines Speicherbereichs (eingegrenzt durch zwei mit einem Punkt voneinander getrennte Speicheradressen) kann als ein Ganzes aufgefaßt werden und mit einem MOVE-Kommando des Monitors von einer Speicherstelle zu einer anderen gebracht werden. Dazu müssen Sie dem Monitor angeben, wo der Speicherbereich liegt und wo er hin soll.
Diese Information besteht aus folgenden Teilen:

- Der Zieladresse,
- einer linken spitzen Klammer (kleiner als),
- der Anfangs- und der Endadresse des Bereichs,
- einem M , damit der Monitor einen Transport (Move) durchführt.

Die Anfangs- und Endadresse geben Sie in gewohnter Weise an (durch einen Punkt getrennt).

Als Beispiel übertragen wir die Speicher 0 - 7 auf 100 - 107, zunächst lassen wir uns diese neuen Speicher ausdrucken:

```
*100.107(RETURN)
 0100: FF FF FF FF FF FF FF FF
*100<0.7M(RETURN)
*100.107(RETURN)
 0100: 6F 3A 01 0B 1A 16 11 07
*
```

Der Monitor kopiert die Werte des angegebenen Bereichs und überträgt sie an den Bestimmungsort. Der Original-Bereich bleibt unverändert. Die Endadresse des Originalbereichs wird jetzt die zuletzt geöffnete Adresse und die nächste veränderbare Adresse ergibt sich aus der Anfangsadresse des Originalbereichs.
Ist die zweite Adresse des angegebenen Bereichs kleiner als die erste, so wird nur

ein Wert (nämlich der Wert der ersten Speicherstelle des Bereichs) übertragen.
Liegt die Zieladresse des MOVE-Kommandos innerhalb des Originalbereichs oder überschneiden sich die beiden Bereiche, so werden die Bereiche speicherweise überschrieben und die Originalwerte der Zieladressen gehen verloren.

Vergleich von zwei Speicherbereichen

Zwei Speicherbereiche können miteinander verglichen werden. Dazu verwenden Sie das selbe Format, wie Sie es soeben beim MOVE-Kommando kennengelernt haben.
Mit dem Vergleichs-Kommando VERIFY läßt sich nach dem MOVE-Kommando feststellen, ob die Übertragung ordnungsgemäß abgelaufen ist.
Das VERIFY-Kommando benötigt wie das MOVE-Kommando eine Zieladresse und einen Bereich.
Der Monitor vergleicht den Inhalt des angegebenen Bereichs mit dem Inhalt des Bereichs ab der Zieladresse.

! Sind die Bereiche gleich, so erfolgt keine Ausgabe.

Sollten Unterschiede erkannt werden, so gibt der Monitor die Adresse mit den jeweiligen unterschiedlichen Inhalten aus.

Beispiel:

```
*100<0.7V(RETURN)            c(1)
*100<0.8V(RETURN)            c(2)
 0008: 00  FF                c(3)
*
```

Herrscht Übereinstimmung wie in (1) (c hier als Kommentar), so erfolgt kein Ausdruck. Im Fall (2) besteht keine Übereinstimmung, es sei denn rein zufällig, deshalb hier der Ausdruck (3).
Beide Adressen bleiben unverändert. Die letzte geöffnete und die nächste veränderbare Adresse ergibt sich jeweils wie im MOVE-Kommando. Wenn die Endadresse kleiner ist als die Anfangsadresse, wird nur die Anfangsadresse verglichen. Auch bei VERIFY kommt es zu Schwierigkeiten, wenn die Zieladresse im Originalbereich liegt.

Programmieren und Starten von Maschinenprogrammen

Viele Programme, die in einer höheren Programmiersprache, wie BASIC oder CP/M geschrieben sind, greifen auf Unterprogramme zu, die in der Maschinensprache eines der auf der Hauptplatine des BASIS 108 untergebrachten Mikroprozessors, des 6502, geschrieben wurden.
Der Monitor hat spezielle Befehle, um den Programmierern, die sich mit der Maschinensprache des 6502 befassen, beim Erstellen von Unterprogrammen zu helfen.

Sie können ein Maschinenprogramm schreiben und die hexadezimalen Werte der

Befehlsteile und der zugehörigen Adressteile mit den oben beschriebenen Kommandos in die Speicherstellen eintragen. Mit Hilfe des Monitor-ROMs können Sie einen hexadezimalen Speicherauszug Ihres Programms erhalten, es überall im Speicher verschieben oder es auf Band schreiben und wieder einlesen. Das wichtigste Kommando im Zusammenhang mit der Maschinensprache ist aber das GO -Kommando (gehen). Wenn Sie eine Speicherstelle öffnen und G tippen, veranlaßt der Monitor den Mikroprozessor an der geöffneten Adresse dieses Programm wie ein Unterprogramm zu behandeln; am Ende der Ausführungen sollte ein RTS -Befehl (Rücksprung aus dem Unterprogramm) stehen, um die Kontrolle wieder dem Monitor zu übergeben.
Die von Ihnen erstellten Programme in Maschinensprache können viele Unterprogramme des Monitors aufrufen. Hier wird ein Programm, das die Zahlen 0 bis 9 auf dem Bildschirm ausgibt, eingegeben und gestartet.

Beispiel:

```
*0:A9 B0 20 ED FD 18 69 1 C9 BA D0 F6 60 (RETURN)
*0.9 (RETURN)
 0000: A9 B0 20 ED FD 18 69 01 C9 BA D0 F6 60 00
*0G(RETURN)
 0123456789
*
```

(Den Befehlssatz des 6502 Mikroprozessors finden Sie im Anhang dieses Handbuches.)

Ein hexadezimaler Speicherauszug des Programmes in Maschinensprache ist nicht einfach zu lesen und die Suche nach Fehlern dadurch erschwert. Darum gibt es im Monitor-ROM ein Kommando, das Maschinenprogramme in Assemblersprache ausgibt. Das bedeutet, daß eine unformatierte Menge von Hexadezimalziffern in einzelne Befehle von 1, 2 oder 3 Byte zerlegt wird. Mit L wird das LIST-Programm des Monitor-ROMs aufgerufen.

Beispiel:

```
*0.DL(RETURN)
 0000:    A9 B0       LDA    #$B0
 0002:    20 ED FD    JSR     $FDED
 0005:    18          CLC
 0006:    69 01       ADC    #$01
 0008:    C9 BA       CMP    #$BA
 000A:    D0 F6       BNE     $0002
 000C:    60          RTS
*
```

Das Maschinenprogramm wurde jetzt in Assemblerform ausgegeben.
Vereinfacht läßt sich sagen, daß in der ersten Spalte die Befehle und in der zweiten bzw. dritten die Operanden stehen, die dann in der vierten bzw. fünften Spalte in der Assemblerform ausgegeben werden. Näheres über das Schreiben und Auswerten von Maschinenprogrammen finden Sie in den entsprechenden Handbüchern über Assembler.

Prüfen und Ändern von Registerinhalten des 6502

Beschäftigen Sie sich intensiver mit dem Monitor ROM und dem 6502 Mikroprozessor, dann wollen Sie sicher einmal eins der internen Register des Prozessors ansehen oder es verändern. Das Monitor ROM reserviert fünf Speicherstellen für die fünf 6502-Register: A, X, Y, P (Prozessorzustand) und S (Stackpointer). Das EXAMINE-Kommando des Monitor ROM's wird durch das Fragezeichen ? ausgelöst und zeigt die Inhalte dieser Adressen an. Die Speicherstelle für das 6502-A-Register ist dann die nächste veränderbare Adresse. Wollen Sie die Werte dieser Speicherstelle ändern, so brauchen Sie nur einen Doppelpunkt und dann die Werte, durch Leerzeichen getrennt, eingeben. Beim nächsten G wird das Monitor ROM erst diese Werte in die echten Register des 6502 laden, bevor es den ersten Befehl Ihres Programms ausführen wird.

Beispiel:

```
*?(RETURN)
 A=88 X=13 Y=D8 P=00 S=B7
*:A B(RETURN)
*?(RETURN)
 A=0A X=0B Y=D8 P=B0 S=F8
*
```

Weitere Monitor-Kommandos

Sie können den Zustand der NORMAL-/INVERSE-Darstellung auf dem Bildschirm durch COUT vom Monitor aus bestimmen. Das INVERSE -Kommando des Monitor ROMs stellt durch Tippen von I auf inverse Ausgabe um, allerdings bleiben die Eingabezeilen in der Normaldarstellung.
Der NORMAL-Zustand wird dann durch das Kommando N wieder hergestellt.

Wenn Sie die von der Firma Apple Computer Inc. verfügbaren Applesoft BASIC ROMs oder Integer BASIC ROMs (siehe dazu in Kapitel 1 -Hauptplatine-) eingesetzt haben, können Sie mit Druck auf die Tasten CTRL und gleichzeitig B den Monitor verlassen, um in die BASIC-Sprache zu gelangen. Auf diesem Wege gehen Ihnen aber alle vorhandenen Programme und Variablen verloren. Sind Sie von BASIC in den Monitor gegangen und wollen Sie wieder zurück ins BASIC, ohne Programm und Variablen zu verlieren, so können Sie das mit Q.

Ein weiteres Kommando ist das PRINTER-Kommando. Mit der Eingabe von nP lenken Sie alle Ausgaben, die normalerweise auf dem Bildschirm erscheinen sollen, auf einen Drucker. n gibt an, in welcher Erweiterungsbuchsenleiste Sie die Steuerkarte für Ihren Drucker eingesetzt haben oder ob Sie eine der auf der Hauptplatine eingebauten Steuerungen für Ihren Drucker benutzen, in der Regel 1.

Beispiel:

```
*1P(RETURN)
*
```

Das Kommando 0P bringt die dann folgende Ausgabe des BASIS 108 wieder auf den Bildschirm.

Das KEYBOARD-Kommando K ersetzt die Tastatur des BASIS 108 durch ein entsprechendes anderes Eingabegerät, das über einen der Erweiterungssteckplätze angeschlossen ist, analoger Gebrauch wie beim P .
Entsprechend schaltet 0K wieder auf die Tastatur zurück.

Wichtig: Obwohl nur Erweiterungsbuchsenleisten von 2 bis 7 auf der Hauptplatine eingebaut sind, schaltet das Kommando 9P die eingebaute serielle Schnittstelle auf 'Ausgabe' und das Kommando 9K auf 'Eingabe' um.

Kleine Hilfen für den Umgang mit dem Monitor

Sie können mehrere Kommandos in einer Eingabe zusammenfassen, solange Sie sie durch Leerzeichen voneinander trennen und nicht mehr als 253 Zeichen eingeben. Die Leerzeichen zählen mit.
Sie können außer dem STORE-Kommando, dem Doppelpunkt : , alle Kommandos in beliebiger Reihenfolge angeben.
Da der Monitor alle Werte nach dem Doppelpunkt in aufeinanderfolgende Speicherstellen ablegt, muß dem letzten Wert des STORE-Kommandos ein Buchstabenkommando folgen. Das NORMAL-Kommando N ist dafür ein gutes Trennzeichen, da es meist keine Veränderung bewirkt und überall verwendet werden kann.
Kommandos mit einem Buchstaben, wie L, R, I und N brauchen nicht mit Leerzeichen von anderen Kommandos getrennt werden.
Erreicht der Monitor bei der Bearbeitung einer Eingabezeile ein Zeichen, das er weder als Hexadezimalzahl noch als gültiges Kommandozeichen erkennen kann, führt er alle Kommandos bis zu diesem Zeichen aus. Dann meldet er über den Lautsprecher den Fehler und ignoriert den Rest der Eingabezeile.
Das MOVE-Kommando kann dazu benutzt werden, eine beliebige Folge von Werten in einen Speicherbereich zu übertragen. Dazu wird diese Folge von Werten an den Anfang des Bereichs geschrieben:

Beispiel:

```
*0:1 2 3(RETURN)
*
```

Dabei kommt es auf die Anzahl der zu wiederholenden Werte an (in diesem Fall sind es drei).
Das MOVE-Kommando bekommt dann eine andere Einteilung:

(Anfangsadresse+Anzahl) (Anfangsadresse).(Endadresse-Anzahl)M

Dieses MOVE-Kommando kopiert die Folge von Werten hinter die Originalfolge, überträgt diese Kopie in die anschließenden Speicherzellen und wiederholt diesen Vorgang, bis der gesamte Bereich gefüllt ist.

Beispiel:

```
*3<0.CM(RETURN)
*0.10(RETURN)
 0000: 01 02 03 01 02 03 01 02 03 01 02 03 01 02 03 01
 0010: 00
*
```

Sie können eine Kommandozeile schreiben, die sich selbst oder einen Teil der Zeile unaufhörlich wiederholt. Dazu fängt der Teil, der sich wiederholt, mit einem Buchstabenkommando ,z.B. N , an und endet mit 34:n, wobei n die hexadezimale Position des Zeichens ist, an dem die Schleife anfängt (für das erste Zeichen ist n=0). Damit diese Schleife funktioniert, muß nach dem Wert für n ein Leerzeichen folgen.

Beispiel:

```
*N 0 2 34:0 N (RETURN)
 0000: 01
 0002: 03
 0000: 01
 0002: 03
 0000: 01
 0002: 03


 .
 (RESET)                         c(SHIFT,SHIFT,CRTL)
*
```

Eine derartige Schleife läßt sich nur durch (RESET) stoppen.

Erzeugen eigener Kommandos

Das USER-Kommando wird durch ein U eingegeben und läßt den Monitor zur Adresse $3F8 springen. In diese Adresse können Sie einen JMP-Befehl einsetzen, der zu dem von Ihnen erstellten Programm oder der gewünschten Adresse springt. Ihr Programm kann so z. B. die Register, die Spezialadressen des Monitors oder die Eingabezeile prüfen. Beispielsweise kann durch U der Lautsprecher angesprochen werden, wenn in $3F8 das Kommando $FF3A steht.

Beispiel:

```
*3F8(RETURN)
 03F8: 4C
*3F8: 4C 3A FF(RETURN)
*3F8L(RETURN)
 03F8: 4C 3A FF    JMP   $FF3A
*U(RETURN)
*                               c(der lautsprecher erklingt).
```

Eventuell werden auch einige Speicher ausgedruckt.

Übersicht über die Monitor-Kommandos

Speicherstellen ansehen

(Adresse)

Gibt den Inhalt einer Speicherstelle aus.

(Anfang).(Ende)

Gibt alle Inhalte zwischen (Anfang) und (Ende) aus.

(RETURN)

Zeigt die Werte von max. 16 Speicherstellen nach der zuletzt geöffneten Adresse an.

Speicherinhalte verändern

(Adress):(Wert)

Speichert (Wert) unter (Adresse) ab.

:(Wert) (Wert)...

Speichert ab der nächsten veränderbaren Adresse die Werte in aufeinanderfolgende Speicherstellen.

Übertragen und Vergleichen

(Ziel) (Anfang).(Ende)M

Kopiert die Werte des Bereichs (Anfang).(Ende) in den Bereich (Ziel) ab.

(Ziel) (Anfang).(Ende)V

Vergleicht die Werte des Bereichs (Anfang).(Ende) mit dem Bereich (Ziel).

Schreiben und Lesen auf Band (nur bei Arbeiten mit 40 Zeichen/ Zeile, siehe auch Anhang G)

(Anfang).(Ende)W

Schreibt die Werte des Bereichs nach einer 10 s-Vorinformation auf Band.

(Anfang).(Ende)R

Liest Werte vom Band in den Speicherbereich (Anfang).(Ende) Druckt ERR im Fehlerfall.

Starten und Ausdrucken von Programmen

(Adresse)G

Läßt den Mikroprozessor 6502 ab (Adresse) das Maschinenprogramm ausführen.

(Anfang).(Ende)L

Läßt ab Anfangsadresse bis Endadresse das Maschinenprogramm in Assemblersprache ausgeben. .(Ende) L läßt weitere Befehle ausgeben.

Verschiedenes

?

Zeigt die Inhalte der 6502-Register an.

I

Setzt INVERSE-Modus.

N

Setzt NORMAL-Modus.

CTRL-B

Startet die Sprache, die im ROM des BASIS 108 verfügbar ist.

Q

Setzt die Sprache fort, die im ROM des BASIS 108 verfügbar ist. Genauer gesagt, das Programm springt auf die Adresse, die in den Speicherstellen (3F2,3F3) angegeben ist.

nP

Bestimmt die Ausgabe zu dem Gerät, dessen Steuerkarte in dem durch n angegebenen Erweiterungssteckplatz sitzt. n=0: dann kommt die Ausgabe auf den Bildschirm zurück.n=1: parallele Schnittstelle, Nummer=9:, serielle Schnittstelle).

nK

Nimmt die Eingabe von dem Gerät an, dessen Steuerkarte in dem durch n angegebene Steckplatz sitzt. n=0: dann wird die Eingabe von der Tastatur erwartet. n=9: serielle Schnittstelle.

U

Springt zu dem Maschinenprogramm ab Adresse $3F8.

Liste ausgewählter Monitor-Unterprogramme

Diese Liste enthält einige nützliche Unterprogramme im Monitor-ROM des BASIS 108. Vor dem Aufruf der Unterprogramme laden Sie die nötigen Speicheradressen oder 6502-Registerinhalte. Der Aufruf erfolgt durch einen JSR-Befehl (Sprung ins Unterprogramm) zu der angegebenen Startadresse des Unterprogramms. Es wird die beschriebene Funktion ausführen und die Register so hinterlassen, wie es jeweils angegeben ist. Der Prozessorstatus (C, Z, N, V) wird im allgemeinen geändert.

$FDED COUT Ausgabe eines Zeichens (Character OUTput).
COUT ist das Standard-Unterprogramm für Zeichenausgabe. Das Zeichen, das ausgegeben werden soll, steht im Akkumulator. COUT ruft das aktuelle Unterprogamm zur Zeichenausgabe auf, dessen Adresse in CSW (Adressen $36 und $37) steht.

$FDF0 COUT1 Ausgabe auf den Bildschirm.
COUT 1 bringt das Zeichen im Akkumulator auf den Bildschirm des BASIS 108. Es wird auf die Ausgabeposition gesetzt und bewegt dann diese Position weiter. Das Zeichen wird mit dem Inhalt der NORMAL-/INVERSE-Speicherstelle modifiziert. Die Steuerzeichen RETURN, Zeilenvorschub und Klingelzeichen werden von COUT 1 ebenfalls behandelt. Das Unterprogramm läßt alle Register intakt.

$FE80 SETINV Setzt den INVERSE-Modus (SET INVerse).
Der INVERSE-Modus für COUT 1 wird gesetzt. Dadurch erscheinen alle Zeichen als schwarze Punkte auf weißem Hintergrund, die dann von COUT 1 ausgegeben werden. Das Y-Register wird auf $7F gesetzt, alle anderen Register bleiben unverändert.

$FE84 SETNORM Setzt den NORMAL-Modus (SET NORMal).
Setzt den NORMAL-Modus für COUT 1. So werden alle Zeichen als weiße Punkte auf schwarzem Hintergrund ausgegeben. Das Y-Register erhält den Wert $FF, alle anderen Register bleiben unverändert.

$FD8E CROUT Gibt ein RETURN aus (Carriage Return OUTput).
CROUT sendet ein RETURN zu dem aktuellen Ausgabegerät.

$FDDA PRBYTE Druckt ein Byte als Hexadezimalzahl.
Dieses Unterprogramm gibt den Inhalt des Akkumulators als Hexadezimalzahl auf das aktuelle Ausgabegerät. Der Inhalt des Akkumulators wird verändert.

$FDE3 PRHEX Druckt eine Hexadezimalziffer (PRint HEXadecimal digit).
Dieses Unterprogramm gibt die unteren vier Bits (Bit 3 bis Bit 0) des Akkumulators als eine Hexadezimalziffer aus. Der Inhalt des Akkumulators wird verändert.

$F941 PRNTAX Druckt A und X als eine Hexadezimalzahl (PRiNT A und X in hexadecimal).
Dieses Unterprogramm gibt die Inhalte des Akkumulators und des X-Registers als vierziffrige Hexadezimalzahl aus. Der Akkumulator enthält die linken zwei Ziffern, das X-Register bestimmt die rechten zwei Ziffern. Der Inhalt des Akkumulators wird verändert.

$F948 PRBLNK Druckt drei Leerzeichen (PRint 3 BlaNK spaces).
Gibt drei Leerzeichen über das Standard-Ausgabegerät aus. Der Akkumulator bekommt den Wert $AO und das X-Register den Wert 0.

$F94A PRBL2 Druckt viele Leerzeichen.
Dieses Unterprogramm gibt 1 bis 256 Leerzeichen zur Standardausgabe. Beim Aufruf bestimmt der Inhalt des X-Registers die Anzahl der Leerzeichen. Ist X=0, so werden 256 Leerzeichen ausgegeben. Beim Ausgang hat der Akkumulator den Inhalt $A0 und das X-Register den Inhalt 0.

$FF3A BELL Ausgabe eines Klingel-Zeichens (BELL).
Dieses Unterprogramm sendet ein Klingel-Zeichen (CTRL G) zu dem aktuellen Ausgabegerät. Der Akkumulator bekommt den Wert $87.

$FBDD BELL1 Abgabe eines Tonsignals aus dem Lautsprecher des BASIS 108.
Dieses Unterprogramm erzeugt ein kurzes 2-Ton Signal. Die Inhalte des Akkumulators und des Y-Registers werden verändert.

$FD0C RDKEY Eingabe eines einzelnen Zeichens.
Dies ist das Unterprogramm für Standard-Zeicheneingabe. Ein blinkender Eingabezeiger erscheint auf dem Bildschirm an der Position des Ausgabezeigers und das Unterprogramm springt zu dem aktuellen Eingabe-Unterprogramm, dessen Adresse in KSW (Adressen $38 und $39).

$FD35 RDCHAR Eingabe eines einzelnen Zeichens oder einer Steuer-Anweisung.
RDCHAR ist ein weiteres Eingabe-Unterprogramm, das Zeichen von der Standardeingabe erhält, aber auch die Tasten des Cursorblockes bis auf die beiden Tasten links und rechts unten.

$FD1B KEYIN Lesen eines Zeichens von der Tastatur.
Dies ist das Unterprogramm für die Eingabe über die Tastatur. Nach Abfrage wartet der BASIS 108 auf einen Tastendruck, eine Zufallszahl wird gebildet. Erfolgt ein Tastendruck, so wird der blinkende Zeiger entfernt und der Tastencode in den Akkumulator gegeben. Falls Zusatztaste oder Zeigertaste gedrückt wurde, so ist im Akkumulator Bit 7=0, sonst 1.

$FD6A GETLN Anforderung einer Eingabezeile mit Bereitschaftszeichen.
Das Unterprogramm GETLN sammelt aus einzelnen Zeichen eine Eingabezeile. Ihre Programme können das Bereitschaftszeichen für GETLN in der Speicherzelle $33 bestimmen. Das Unterprogramm GETLN kehrt mit der Eingabezeile im Eingabepuffer (ab Adresse $200) und mit der Länge der Eingabezeile im X-Register zurück. Die Tasten des Cursorblockes werden ausgeführt, die Zusatztasten dagegen nicht.

$FD67 GETLNZ Anforderung einer Eingabezeile.
Das Unterprogramm GETLNZ schickt erst einen Zeilenvorschub zum Standardausgabegerät, bevor GETLN ausgeführt wird (s. oben).

$FD6F Anforderung einer Eingabezeile ohne Bereitschaftszeichen.
Dieser Einsprung beginnt in GETLN erst an der Stelle, an der die Eingabezeile gebildet wird, so daß kein Bereitschafts zeichen erscheint. Löschen Sie jedoch mehr Zeichen als in der Eingabezeile vorhanden waren oder betätigen Sie CE, so wird der Inhalt der Speicherzelle $33 als Bereitschaftszeichen einer neuen Eingabezeile ausgegeben.

$FCA8 WAIT Warten.
Dieses Unterprogramm wartet eine bestimmte Zeit und kehrt dann wieder zu dem Programm zurück, das es aufgerufen hat. Der Akkumulator bestimmt diese Zeit. Wenn A der Inhalt des Akkumulators ist, ergibt sich eine Verzögerung von (13 + 12A + 5A*A)Zyklen. Das ist ca. 1 Mikrosekunde. Bei A = 0 zählt es wie 256. WAIT läßt X und Y unverändert, nur das A-Register wird 0.

$F864 SETCOL Setzt die Farbe für die Ausgabe von Lo-Res Graphik (SET COLor).
Der Akkumulator bestimmt die Farbe, die bei der Lo-Res Graphik-Ausgabe auf den Bildschirm verwendet werden soll. Der Akkumulator wird verändert, sonst ändern sich die Register nicht.

$F85F NEXTCOL Die Farbnummer wird um 3 erhöht (NEXT COLor).
Die aktuelle Farbe für die Ausgabe von Lo-Res Graphik wird um 3 erhöht. Nur das A-Register wird verändert.

$F800 PLOT Überträgt einen Block auf den Lo-Res Bildschirm.
Dieses Unterprogramm druckt einen einzelnen Block in der vorher eingestellten Farbe auf den Bildschirm, beim 80 Spalten Monitor-ROM bis zu 79 Zeichen. Die vertikale Position wird im Akkumulator übergeben und die horizontale Position wird dem Y-Register entnommen. PLOT verändert nur den Akkumulator.

$F819 HLINE Zeichnet eine waagrechte Linie von Blöcken.
Es wird eine Zeile von Blöcken in der vorher festgelegten Farbe auf den Lo-Res-Bildschirm gezeichnet (s. auch PLOT). Folgende Angaben müssen beim Aufruf vorhanden sein: Die senkrechte Koordinate steht im Akkumulator, die waagrechte Koordinate des linken Endes im Y-Register, die des rechten Endes in $2C. HLINE verändert A und Y, läßt aber X intakt.

$F828 VLINE Zeichnet eine senkrechte Linie von Blöcken.
Dieses Unterprogramm zeichnet eine senkrechte Linie von Blöcken der vorher festgelegten Farbe auf den Lo-Res-Bildschirm. Folgende Werte müssen beim Aufruf vorliegen:
Die oberste vertikale Position im Akkumulator, die unterste vertikale Koordinate in $2D und die horizontale Koordinate der Linie im Y-Register. VLINE verändert den Akkumulator.

$F832 CLRSCR Löscht den gesamten Lo-Res Bildschirm.
CLRSCR löscht den gesamten Bildschirm der Blockgraphik. Wird CLRSCR im TEXT-Modus aufgerufen, so wird der Bildschirm mit inversen §-Zeichen gefüllt. CLRSCR verändert die Inhalte von A und X.

$F836 CLRTOP Löscht den oberen Teil der Lo-Res Graphik.
CLRTOP arbeitet wie CLRSCR (s. oben), aber es werden nur die oberen 40 Reihen des Bildschirms gelöscht.

$F871 SCRN Liest ein Zeichen auf dem Lo-Res Bildschirm.
Dieses Unterprogramm kehrt mit der Farbe eines bestimmten Blocks auf dem Bildschirm in das Programm zurück, das SCRN aufgerufen hat. Den Anruf gestalten Sie wie bei PLOT (s. oben). Die Nummer der Farbe des Blocks steht nach dem Aufruf im Akkumulator. Andere Register werden nicht verändert.

$FB1E PREAD Liest die Stellung einer Spielsteuerung.
PREAD braucht zum Aufruf die Nummer der Spielsteuerung im X-Register. Diese Zahl muß 0, 1, 2 oder 3 sein, sonst werden Sie sich wundern. Die Stellung der Spielsteuerung wird als Zahl zwischen $00 und $FF im Y-Register übergeben. Der Akkumulator wird verändert.

$FF4A SAVE Rettet alle Register.
Die Inhalte aller internen Register des 6502-Mikroprozessors werden in der Reihenfolge A-X-Y-P-S in die Speicherstellen $45 bis $49 geschrieben. Die Inhalte von A und X werden verändert und der Dezimalmodus des Mikroprozessors wird gelöscht.

$FF3F RESTORE Register werden wiederhergestellt.
Die Inhalte der internen Register des 6502-Mikroprozessors werden von den Speicherstellen $45 bis $48 geladen. S (stack) Register wird nicht geändert, damit Restore zurückkehren kann.

SPEZIALADRESSEN DES MONITORS

Adresse Dezimal	Hexa	Verwendung im BASIS 108 Monitor
1008 1009	$3F0 $3F1	Enthält die Adresse des Unterprogramms, das "BRK"-Befehle behandelt (normal: $FA59).
1010 1011	$3F2 $3F3	Warmstart in die benutzte Sprache. Monitor "Q" springt auf die Adresse.
1012	$3F3	Einschalt-Byte
1013 1014 1015	$3F5 $3F6 $3F7	Enthält einen JMP (Sprung)-Befehl zu dem Unterprogramm, das FPBASIC -Kommando behandelt . (Normal: $4C $58 $FF)
1016 1017 1018	$3F8 $3F9 $3FA	Enthält einen JMP-Befehl zu dem Unterprogramm, das "USER" (U)-Kommandos behandelt.
1019 1020 1021	$3FB $3FC $3FD	Enthält einen JMP-Befehl zu dem Unterprogramm, das nichtmaskierbare Interrupts behandelt.
1022 1023	$3FE $3FF	Enthält die Adresse des Unterprogramms, das maskierbare Interrupts (IRQ) behandelt.
1273	$4F9	Wenn 0, dann 40 Zeichen, wenn ≠ 0, dann 80 Zeichen.

KAPITEL 5

INHALTSVERZEICHNIS

Der Speicher

Speicherorganisation

Das BASIS 108 Computersystem kann mit einem RAM-Speicher bis zu 128 kByte ausgerüstet werden. Der 6502 Mikroprozessor (wie auch der Z-80 Mikroprozessor) kann allerdings mit seinen 16 Adressleitungen nur einen Speicherraum von 64 kByte verwalten. Zusätzlich zu dem RAM-Speicher ist ein ROM-Bereich von 12kByte und der Ein-/Ausgabebereich, der einen Adressraum von 4 kByte belegt, zu adressieren. Da sich somit ein Adressraum von 144 kByte ergibt, den es zu adressieren gilt, wurde die Möglichkeit geschaffen, nur bestimmte Teile des ROM- und RAM-Bereiches zur gleichen Zeit zu aktivieren.
Um dies zu erreichen, wurde der RAM-Bereich zunächst in 2 Seiten, Banks genannt, von je 64 kByte Größe eingeteilt, dann jeder Bereich nochmals in 8 kByte Blöcke. Dadurch besteht die Möglichkeit, zwischen den Banks in Schritten von 8 kByte umzuschalten. Der nächste Schritt war nun, den ROM-Bereich in den Adressraum zu integrieren. Da der 6502 Mikroprozessor nach einem Reset die Adresse $FFFC ausgibt und auf dieser eine ausführbare Operation ständig gespeichert sein muß, ist der ROM-Bereich am Ende des Adressraumes angesiedelt, dem sich direkt der Ein-/Ausgabebereich anschließt.

Aufteilung des Adreßraumes

Adresse		BANK 0		BANK 1	
$FFFF		LC0		LC1	
$DFFF	ROM	LC00	LC01	LC10	LC11
$D000		I/O*		I/O*	
$C000					
$6000		HGR 2		RAM	
$4000		HGR 1			
$2000					
$0BFF	80Z	TEXT2			
$0800	80Z	TEXT1			
$0400					
$0200		STACK0		STACK1	
$0100		ZERO P0		ZERO P1	
$0000					

* I/O-Ein-/Ausgabe
Damit haben wir die oberen 16 kByte des Adressraumes einmal mit ROM und Ein-/Ausgabebereich belegt und zum anderen existiert auch noch der RAM-Speicher für diesen Bereich. Dieser 16 kByte große Speicher wird noch einmal in 4 kByte Blöcke augeteilt. Da der 4 kByte Ein-/Ausgabebereich dem Prozessor ständig zur

Verfügung stehen muß, wird der für diesen Adressraum vorgesehene RAM-Speicher dem nächsten 4 kByte Block parallel geschaltet. Die Wahl, welcher dieser beiden Blöcke nun aktiv sein soll, kann dann über einen Software-Schalter getroffen werden (s. unten). Da dieser RAM-Speicher parallel zum ROM-Speicher liegt und nur ein Bereich aktiv sein darf, wird auch hier der aktive Bereich durch einen Softwareschalter ausgewählt.
Um diesen RAM-Bereich für besondere Aufgaben einsetzen zu können (z. B. Speicherung eines Basic Interpreters o. ä.) ist es möglich, diesen Bereich vor unbeabsichtigtem Schreiben zu schützen. Auch ist eine Kombination von ROM-Lesen und RAM-Schreiben möglich.
All diese oben genannten Möglichkeiten werden über Softwareschalter erreicht und gelten sowohl für die BANK 0 als auch für die BANK 1.

Im RAM-Bereich der BANK 0 sind außerdem die verschiedenen Bereiche der Bildwiederholungsspeicher angesiedelt. Eine Darstellung der Bildwiederholungsspeicher in der BANK 1 ist nicht möglich, da bei einem Speicherzugriff der Bildwiederholungslogik immer BANK 0 durch die Hardware verwendet wird.
Den beiden Textseiten des Bildwiederholungsspeichers ist ein 2kByte statisches RAM parallel geschaltet, um die 80 Zeichen pro Zeile Darstellung zu ermöglichen. Wenn nun in den Bildwiederholungsspeicher Nr. 0 Zeichen geschrieben werden sollen, wird je nach Position dieses Zeichens, entweder der RAM-Bereich des normalen RAM's oder das statische RAM aktiviert.

BANK 0/BANK 1 - Umschaltung

Die nachfolgenden Adressen schalten zwischen BANK 0 und BANK 1 um.
Die Umschaltung erfolgt aber nur, wenn ein Schreibbefehl auf diese Adresse ausgeführt wird. Ein Lesebefehl dieser Adressen liest den Zustand der entsprechenden TTL- und Analogeingänge.
Nach dem Einschalten des BASIS 108 Computersystems oder einem RESET ist grundssätzlich die BANK 0 aktiv.

Bank 0 aktiv	Bank 1 aktiv	Adressraum
$C060w	$C061w	$0000 - $1FFF
$C062w	$C063w	$2000 - $3FFF
$C064w	$C065w	$4000 - $5FFF
$C066w	$C067w	$6000 - $7FFF
$C068w	$C069w	$8000 - $9FFF
$C06Aw	$C06Bw	$A000 - $BFFF
$C06Cw	$C06Dw	$D000 - $DFFF
$C06Ew	$C06Fw	$E000 - $FFFF

Der Schalter $C06C/$C06D schaltet nur den 4 kByte Adressraum von $D000 bis $DFFF, der Adressraum $C000 bis $CFFF ist der Ein-/Ausgabebereich und kann daher nicht geschaltet werden.

ROM und RAM Umschaltung

Die nachfolgend beschriebenen Schalter erlauben die Umschaltung zwischen ROM und RAM der jeweils aktivierten BANK im Adressbereich $E000-$FFFF, sowie das Umschalten des mit RAM-Speicher doppelt belegten Adressbereichs $D000 bis $DFFF und das Schützen dieser Bereiche vor versehentlichem Beschreiben. Die Schaltergruppe $C080 bis $C083 bezieht sich auf den Block LCx0 und die Gruppe $C088 bis $C08B auf die Blöcke LCx1, wobei x durch die jeweils aktivierte Bank dargestellt wird, (Bank 0 x=0; Bank 1 x=1).
Die nachfolgenden Schalteradressen sollen nur durch Leseoperationen angesteuert werden.

RAM-Auswahl $D000 - $DFFF Seite 0/Seite 1		RAM/ROM-Auswahl
$C080	$C088	RAM ist schreibgeschützt, Lesen erlaubt, ROM ist abgeschaltet.
$C081	$C089	ROM Lesen erlaubt, RAM schreibgeschützt. Wird der Befehl zwei oder mehrmal gegeben, ist es möglich im RAM zu schreiben.
$C082	$C08A	RAM schreibgeschützt, es wird aus ROM gelesen.
$C083	$C08B	erlaubt den RAM zu lesen, schreibgeschützt. Wird der Befehl zwei- oder mehrmal gegeben, so kann auch geschrieben werden.

Einige Erklärungen zu den Schaltern:

$C080/$C088 Der RAM-Bereich wird nur für Leseoperationen aktiviert und der ROM-Bereich abgeschaltet.

$C081/$C089 Der ROM-Bereich wird für Leseoperationen aktiviert und der RAM-Bereich hierfür abgeschaltet. Bei zwei- oder mehrmaligem Ansprechen wird der RAM-Bereich für Schreiboperationen aktiv, so daß zum Beispiel das Kopieren der ROMs in den RAM-Bereich möglich ist.

$C082/$C08A Schaltet das RAM Lesen ab und aktiviert den ROM-Bereich. Der RAM-Bereich bleibt aber schreibgeschützt.

$C083/$C08B Der RAM-Bereich wird für Leseoperationen aktiviert. Bei zwei- oder mehrmaligem Ansprechen wird der RAM auch schreibfähig. Das bedeutet, daß dieser Bereich nun ein normales RAM-Memory darstellt.

Das Statik-RAM für die 80 Z-Darstellung

Dieses 2K Statik-RAM ist dem Adressbereich $0400-$0BFF parallel geschaltet. Dies ermöglicht 2 Seiten Bildschirmwiederholungsspeicher mit je 80 Zeichen pro Zeile bei 24 Zeilen. Da auch dieser Bereich parallel zum normalen RAM-Bereich liegt, wird über einen Softwareschalter der jeweilig aktive Bereich ausgewählt.
$C00Dw Zusatz RAM eingeschaltet, Normal RAM abgeschaltet
$C00Cw Zusatz RAM abgeschaltet, Normal RAM eingeschaltet.

Diese Softwareschalter sind nur mit einem Schreibbefehl zu betätigen.

Kapitel 6

INHALTSVERZEICHNIS

Ein-/Ausgabe

Eingebaute Ein-/Ausgabemöglichkeiten

Auf der Hauptplatine des BASIS 108 sind folgende Ein- und Ausgabemöglichkeiten integriert:

- Paralleles Drucker Interface (Centronics kompatibel),
- Serielles RS 232c Interface,
- Kassettenrekorder Interface,
- Anschluß für 4 Handregler,
- 3 Eingänge für TTL-Signale,
- 4 TTL-Ausgänge,
- Lautsprecherausgang,
- Tastatur,
- Video.

Man kann diese Ein- und Ausgabemöglichkeiten in mehrere Gruppen einteilen; Dateneingänge, Strobes, Softwareschalter, Kippschalter und Statuseingänge.

Dateneingänge

Als Dateneingänge des BASIS 108 Systems kann neben der parallelen und seriellen Schnittstelle auch der Tastatureingang gewertet werden. Das höchstwertige Bit dieses Einganges ist ein Statusbit und die niederwertigen 7 Bits der entsprechenden ASCII-Code der gedrückten Taste. Ist das höchstwertige Bit 1, wurde auf der Tastatur eine Taste gedrückt.

Status Eingänge

Diese Eingänge können nur die Zustände EIN oder AUS annehmen. Angezeigt wird dieses im höchstwertigsten Bit der angesprochenen Adresse. Das Erkennen des entsprechenden Zustandes kann von einer höheren Programmiersprache durch Testen des gelesenen Bytes, ob größer oder gleich 128 für EIN und kleiner als 128 für AUS durchgeführt werden. Solche Eingänge sind die 3 TTL-Eingänge, der Kassettenrekorder Eingang und die Handreglereingänge.

Strobe

Signale dieses Typs werden ebenfalls über Speicheradressen erzeugt und dienen zum definierten Setzen oder Rücksetzen einiger Statuseingänge. Im BASIS 108 Computersystem existieren 3 Strobe Signale.

1. Tastatur Strobe ($C010), dieses Strobe Signal setzt das höchstwertigste Bit des Tastatureinganges ($C000) auf NULL zurück.

2. Der Handregler Strobe ($C070) setzt alle vier Mono-Flops der Handreglereingänge zurück und startet die Zeitschleife neu.

3. Der Utilitie Strobe ($C040) ist auf Pin 5 des Handregleranschlusses zu finden. Wenn diese Adresse angesprochen wird, geht diese Leitung für 0.4 Mikrosekunden von TTL-high auf TTL-low. Wenn mit einem Schreibbefehl der Form absolut-indiziert oder indirekt- indiziert diese Adresse angesprochen wird, werden 2 Pulse erzeugt. Wenn der 6502 Mikroprozessor einen Schreibbefehl ausführt, liest er zuerst die angesprochene Adresse, bevor sie überschrieben wird. Dadurch erfolgen bei einem Schreibbefehl zwei Zugriffe zu der entsprechenden Adresse.

Kippschalter

Der Lautsprecher, wie auch der Kassettenrekorder-Ausgang werden über einen Kippschalter angesprochen.
Ein Lesen der entsprechenden Adresse veranlaßt ein Flip-Flop in den anderen Zustand zu fallen. Das bedeutet; der Ausgang des Flip-Flops geht von logisch 0 auf logisch 1 und bleibt solange in diesem Zustand, bis das Flip-Flop erneut angesprochen wird.

Drucker Interface

Das parallele Drucker Interface generiert alle notwendigen Signale zur Steuerung eines Druckers mit Centronics kompatibler Schnittstelle. Die Ausgabedaten werden in die Ausgabeadresse $C090-C097 geschrieben, wodurch automatisch die Generierung eines Strobe Signals ausgelöst wird. Im höchstwertigen Bit der Adresse $C1C1 kann die Übernahmebestätigung (Acknowledge) des Druckers abgefragt werden. Eine Standard Treiber Routine ist in einem 256x8 ROM auf der Adresse $C100 abgelegt.

Serielles RS 232c Interface

Das serielle Interface besteht aus dem Baustein 6551 mit nachgeschalteten Leitungsempfängern und Treibern. Dieser Baustein hat 2 Handshakeleitungen. Das Datenregister dieses Bausteins ist auf der Adresse $C098, das Statusregister auf $C099, das Command Register auf $C09A und das Mode Register auf der Adresse $C09B. Die Übertragungsgeschwindigkeit kann zwischen 50 und 19200 Baud gewählt werden. Eine Standard Treiber Routine befindet sich ebenfalls in dem ROM auf der Adresse $C108. Diese Treiber Routine initialisiert das serielle Port auf folgende Werte:

9600 Baud, Wortlänge 8 Bit und 2 Stopbit, keine Parität.

Wollen Sie die V24 Treibersoftware oder andere Parameter benutzen, schlagen Sie bitte im Anhang E nach. Auf den nachfolgenden Seiten finden Sie hierfür die wichtigsten Parameter dieses Bausteines.

Adressen	Schreiben	Lesen
$C098	Transmit Data Register	Receiver Data Register
$C099	Programm Reset *	Statusregister
$C09A	Comm. Register	
$C09B	Contr. Register	

* Ein Schreiben auf die Adresse des Statusregisters bewirkt ein Setzen des ACIA in einen bestimmten Status. Hiervon werden alle Register betroffen (für weitere Informationen s. Datenblatt im Anhang).

Kontrollregister

Mit dem Kontrollregister wird die Wortlänge, die Anzahl der Stopbits und die Übertragungsrate festgelegt.

Bit 7 — STOP BITS
0 = 1 Stopbit
1 = 2 Stopbits
1 Stopbit, wenn die Wortlänge 8 und Parität gesetzt ist.
1,5 Stopbits, wenn die Wortlänge 5 und keine Parität gesetzt ist.

Bit 6 u. 5 — Wortlänge

0 0 8 Bit
0 1 7 Bit
1 0 6 Bit
1 1 5 Bit

Bit 4 — Empfänger Takt Frequenz
1 = Interner Baud Rate Generator
! muß immer 1 sein !

Baud Rate Generator
-mit diesen Bits wird die Baud Rate
ausgewählt-

Bit 3 2 1 0	Baud Rate
0 0 0 0	illegal
0 0 0 1	50 Baud
0 0 1 0	75
0 0 1 1	110
0 1 0 0	134,5
0 1 0 1	150
0 1 1 0	300
0 1 1 1	600
1 0 0 0	1200
1 0 0 1	1800
1 0 1 0	2400
1 0 1 1	3600
1 1 0 0	4800
1 1 0 1	7200
1 1 1 0	9600
1 1 1 1	19200

Kommandoregister

Das Kommandoregister steuert spezielle Sende- und Empfangsfunktionen.

Überprüfung der Paritäten

Bit 7	6	5	
x	x	0	keine Parität bei Sendung und Empfang
0	0	1	ungerade Sender und Empfänger
0	1	1	gerade Sender und Empfänger
1	0	1	Sendet 1 statt Parität Parität Test abgeschaltet
1	1	1	Sendet 0 statt Parität Parität Test abgeschaltet.

Bit 4 Normal/Echo Mode Empfänger

0=Normal
1=Echo

Transmitter Kontrolle

Bit	3	2	Transmitter Unterbrechung	RTS Pegel
	0	0	abgestellt	inaktiv
	0	1	eingeschaltet	aktiv
	1	0	abgestellt	aktiv
	1	1	abgestellt	aktiv, es wird BREAK gesendet.

Bit 1 Empfangsunterbrechung
0 = eingeschaltet
1 = ausgeschaltet.

Bit 0 Data Terminal Ready (DTR)
0 = Empfang aus / Baustein (DTR inaktiv)
1 = Empfang an / Baustein (DTR aktiv)

Statusregister

Im Statusregister wird der aktuelle Zustand des Bausteins angezeigt.

Bit 7 Interrupt (IRQ)
0 = kein Interrupt
1 = Interrupt ist aufgetreten

Bit 6 Data Set Ready (DSR)
0 = DSR bereit
1 = DSR nicht bereit

Bit 5 Data Carrier Detect (DCD)
0 = DCD erkannt
1 = DCD nicht erkannt

Bit 4 Datensenderegister
0 = nicht leer
1 = leer

Bit 3 Datenempfangsregister
0 = nicht voll
1 = voll

Bit 2 Überlauf
0 = kein Fehler
1 = Fehler, Datenverlust, da nicht schnell genug gelesen.

Bit 1 Taktfehler
0 = kein Fehler
1 = Fehler, wahrscheinlich falsche Baudrate

Bit 0 Paritätsfehler
0 = kein Fehler
1 = Fehler wurde erkannt

Kasettenrekorder Interface

Das Einlesen einer Information vom Kasettenrekorder geschieht auf der Adresse $C060, die Ausgabe auf $C02x. Eine entsprechende Treiberroutine ist im speziellen Monitor-ROM für 40 Zeichen/Zeile untergebracht. Dieses Monitor-ROM muß gesondert erworben werden, s. Anhang G.

Handregleranschluß und TTL Ein- und Ausgänge

Der Handregleranschluß und die TTL Ein- und Ausgänge sind gemeinsam auf einem 16-poligen DIL-Sockel verfügbar.
Über den Regelwiderstand des Handreglers wird die Rücksetzzeit eines monostabilen Flip-Flops gesteuert. Das Setzen oder Starten aller 4 Flip-Flops wird über die Adresse $C07x gesteuert, die Abfrage des Status der einzelnen Flip-Flops auf den Adressen $C064 bis $C067.
Die 4 TTL-Ausgänge sind auf den Adressen $C058 bis $C05F und die 3 TTL-Eingänge auf den Adressen $C061 bis $C063.
Auf dem DIL-Sockel befindet sich noch ein weiteres Signal, welches über die Adresse $C04x angesprochen wird und dem Benutzer zur freien Verfügung steht.

Lautsprecher

Durch Ansprechen der Adresse $C03x wird ein Flip-Flop geschaltet und der Lautsprecher erzeugt ein einmaliges Klick-Geräusch. Durch ein entsprechendes Programm lassen sich Töne verschiedenster Frequenzen und Dauer produzieren.

Erweiterungs-ROM

Das BASIS 108 Computersystem besitzt 6 Erweiterungssteckplätze für Interfacekarten oder andere Erweiterungskarten. Um diese Steckplätze vorteilhaft ausnutzen zu können, sind jedem Steckplatz 2 direkte Adressbereiche und allen gemeinsam zusätzlich noch ein 2 KByte großer Adressraum zugeordnet. Im einzelnen gleichen sich diese Bereiche wie folgt:

1. Peripheriekarten I/O Adressen.
   Dies sind 16 Adressen für jeden Steckplatz. Die Signalleitung DEVICE SELECT (PIN 41 jedes Steckplatzes) signalisiert, daß der Prozessor eine Adresse innerhalb dieses Bereiches anspricht. Diese Adressen sollten bevorzugt für Ein-/Ausgabe Operationen verwendet werden.

Peripheriekarte I/O Zuweisung
x = $0 . .$F

$C0Ax		2
$C0Bx		3
$C0Cx	Ein/Ausgabe für Steck-	4
$C0Dx	platznummer	5
$C0Ex		6
$C0Fx		7

2. Peripheriekarten ROM Adressraum.
   Ein weiterer Adressraum von 256 Byte ist jedem Steckplatz für die Aufnahme von Treiberroutinen oder ähnlichem direkt zugeordnet.
   Die I/O SELECT Leitung (Pin 1 jedes Steckplatzes) zeigt, wenn sie auf logisch 0 geht, daß eine Adresse in diesem Bereich angesprochen wird.
   Die Startadresse eines jeden Steckplatzes ergibt sich direkt aus der Nummer des Platzes. Steckplatz 3 hat die Startadresse $C300 (im hexadezimalen Format).

Peripheriekarte PROM Zuweisung
xx = 00 . FF

$C2xx		2
$C3xx		3
$C4xx	PROM Raum für Steck-	4
$C5xx	platznummer	5
$C6xx		6
$C7xx		7

Der Adressraum von $C800 bis $CFFF ist einem 2 KByte Erweiterungs-ROM oder EPROM vorbehalten. Dieser Bereich ist nur einmal vorhanden und das ROM sollte über eine Selektionslogik auf den Peripheriekarten aktiviert werden.
Das Signal I/O STROBE (PIN 20 eines jeden Steckplatzes) zeigt an, daß der Prozessor auf eine Adresse dieses Bereiches zugreifen möchte.
Auf jeder eingesetzten Peripheriekarte kann ein ROM für diesen Adressraum installiert sein, aber nur jeweils ein ROM darf aktiv sein. Um dies zu erreichen, sollte die Aktivierung des ROMs über ein R-S Flip-Flop gesteuert werden. Der Setzeingang des Flip-Flops sollte durch eine definierte Adresse des I/O SELECT angesteuert und mit der Adresse $CFFF zurückgesetzt werden. Die Adresse $CFFF sollte zur Deaktivierung des ROMs oder EPROMs immer benutzt werden. Nach Benutzung dieses Bereiches sollte durch $CFFF ein eventuell aktives ROM oder EPROM abgeschaltet und anschließend gezielt das neue ROM oder EPROM aktiviert werden. Eine entsprechende Routine kann in dem 256 Byte Adressraum des entsprechenden Steckplatzes abgelegt sein. Ein großer Vorteil dieses Adressbereiches ist, daß bei der Erstellung der Software für diesen Bereich nicht auf Verschiebbarkeit der Software geachtet werden muß, da das ROM unabhängig vom

Steckplatz immer auf den Adressen $C800 bis $CFFF liegt.

Beispiel:

```
BIT   $CFFF ;  Abschalten aller C8-ROMs,
BIT   $C300 ;  Einschalten des C8-ROM von Slot 3,
LDA   #$C3
STA   $7F8
JSR   $C800 ;  Benutzung der C8-ROMs.
```

Da es für viele Aufgaben zweckmäßig ist, neben dem ROM auch einen RAM-Bereich für die Peripheriekarte zur Verfügung zu haben, werden RAM-Adressen, die durch den Bildwiederholungsspeicher nicht benutzt werden, den einzelnen Steckplätzen zugeordnet.

I/O RAM Zwischenspeicher

Basis-	Steckplatznummer						
adressen	1*	2	3	4	5	6	7
$0478	$0479	$047A	$047B	$047C	$047D	$047E	$047F
$04F8	$04F9	$04FA	$04FB	$04FC	$04FD	$04FE	$04FF
$0578	$0579	$057A	$057B	$057C	$057D	$057E	$057F
$05F8	$05F9	$05FA	$05FB	$05FC	$05FD	$05FE	$05FF
$0678	$0679	$067A	$067B	$067C	$067D	$067E	$067F
$06F8	$06F9	$06FA	$06FB	$06FC	$06FD	$06FE	$06FF
$0778	$0779	$077A	$077B	$077C	$077D	$077E	$077F
$07F8	$07F9	$07FA	$07FB	$07FC	$07FD	$07FE	$07FF

*Diese Adressen werden von den eingebauten seriellen und parallelen Treibern schon benutzt.

A N H A N G

INHALTSVERZEICHNIS

A N H A N G   A

HINWEISE ZUR SOFTWARE-KOMPATIBILITÄT MIT APPLE II

Die ZAP:-Diskette erfüllt drei verschiedene Funktionen:

1. Modifizierung des Apple-Pascal 1.1-Systems, so daß die 80-Zeichendarstellung und die eingebaute Parallel- und Seriellschnittstelle verfügbar sind.

2. Modifizierung des Microsoft CP/M-Systems, um ebenfalls die 80-Zeichendarstellung und die Schnittstellen verfügbar zu machen.

3. Laden der gewünschten BASIC-Version.

Das Herstellen dieser Modifizierungen brauchen Sie nur einmal durchzuführen, mit den geänderten Disketten können Sie dann arbeiten, wie in anderen Systemen auch üblich. Siehe auch Kapitel 2 und die entsprechenden Betriebshandbücher.

Die Beschreibung für diese Operationen setzt zwei Laufwerke voraus. Bei nur einem Laufwerk bitten Sie Ihren Händler um Hilfe beim Anpassen der Disketten.

Zu 1. **Hinweise zur Anpassung des Apple-Pascal 1.1 Systems**

Um die gewünschte Pascalversion zu erhalten, müssen die Files SYSTEM.APPLE und SYSTEM.MISCINFO, die sich auf der Diskette APPLE1: befinden, verändert werden.

Verwenden Sie für die Anpassung eine Kopie Ihrer Apple-Pascal Diskette, nicht das Original.

Im folgenden werden im Text die Abläufe intern und extern beschrieben . Dann folgen die Ein- und Ausgaben auf dem Bildschirm. Dabei sind Ihre Eingabebefehle gesperrt gedruckt und die Ausgaben in Großschreibung ausgeführt. Nur die zu drückende Returntaste ist bei Ihren Eingaben als (RETURN) angegeben.

Transferieren Sie zunächst wie folgt das File SYSTEM.APPLE von der Diskette APPLE1: auf die Diskette ZAP: . Stecken Sie die Diskette APPLE1: in Laufwerk 1, die Diskette ZAP: in Laufwerk 2 und schalten Sie den Rechner ein. Durch Drücken der Taste F gelangen Sie in den Filer. Rufen Sie nun die Transferroutine durch Drücken der Taste T auf:

```
      F
FILER: G, S, W, N, L, R, C, T, D, Q
      T
WHAT FILE ? APPLE1:SYSTEM.APPLE (RETURN)

TO WHERE ?  ZAP:$ (RETURN)
      Q
```

Durch das Drücken von Q gelangen Sie wieder zur Kommandozeile. Es geht weiter mit dem Drücken der Taste X :

```
      X
EXECUTE WHAT FILE ? ZAP:ZAP (RETURN)
VERSION 2.0 ZAP, 29-MARCH-82          (das datum muß nicht
(C) SANDOR SCARI 1982                  identisch sein)
BUFFER SIZE: 54 BLOCKS

COMMAND CONSOLE: COMMAND 'ZAP:PASCAL' (RETURN)
```

Das Programm ZAP legt jetzt auf der Diskette Zap: eine modifizierte Version des Files SYSTEM.APPLE unter dem Namen NEW.APPLE ab. Während das Programm arbeitet, läuft das Laufwerk, in dem sich die ZAP:-Diskette befindet. Außerdem erscheinen verschiedene Texte auf dem Bildschirm. Nach Beendigung des Programms erscheint die Kommandozeile auf dem Bildschirm.
Die Files NEW.APPLE und 108.MISCINFO müssen nun von der Diskette ZAP: auf die Diskette APPLE1: mit folgenden Kommandos transferiert werden:

```
       F
FILER: G, S, W, N, L, C, T, D, Q
       T
TRANSFER WHAT FILE ? ZAP:NEW.APPLE (RETURN)
TO WHERE ? APPLE1:SYSTEM.APPLE (RETURN)
REMOVE OLD SYSTEM.APPLE ? Y

       T
TRANSFER WHAT FILE ? ZAP:108.MISCINFO (RETURN)
TO WHERE ? APPLE1:SYSTEM.MISCINFO (RETURN)
REMOVE OLD SYSTEM.MISCINFO ? Y
```

Hiermit ist die Prozedur der Änderung der Diskette APPLE1 für das Apple-Pascal beendet.

Laden Sie Ihr System neu. Wenn Sie die Reihenfolge eingehalten haben und alle Operationen richtig ausgeführt haben, arbeitet Ihr Apple Pascal 1.1 jetzt mit 80 Zeichen/Zeile.

Im folgenden sind einige Zeichen aufgeführt, die durch die Änderung der Diskette anders sind.

1. Editor-Accept ist die **'HOME'** Taste des Cursorblocks, bei Apple CRTL-C.

2. Die Pfeiltaste ' ⇐ ' entspricht der Apple-Taste 'Pfeil links', die Pfeiltaste ' ⇒ ' der Apple-Taste 'Pfeil rechts. Die Pfeiltasten 'rechts, links, oben und unten' werden vom Editor richtig gedeutet und ausgeführt.
3. Die Zusatztaste **Shift-CTRL-F15** ist mit BREAK belegt.

4. Die Zusatztaste **Shift F1** ist mit Stop belegt. (Hält die Ausgabe an).

5. Die Zusatztaste **Shift F2** ist mit Flush belegt. (Bildschirmausgabe wird unterdrückt).

Alle anderen Zusatztasten können Sie frei verwenden (Zusatztasten sind daran zu erkennen, daß Bit 7 gesetzt ist, d. h. ASCII über 127).

Hier ein Auszug aus einem entsprechenden Abfrageprogramm:

```
Read(Keyboard,ch);
if ord(ch) ≥128 then
writeln('Funktion' ord(ch):4).
```

zu 2) **Anpassung von Apple CP/M-Disketten an den BASIS 108**

Ziel der Anpassung ist es, die 80-Zeichen-Darstellung, das Parallelinterface und die V24-Schnittstelle des BASIS 108 unter CP/M nutzen zu können.

Wie im vorigen Abschnitt sind die Ausgaben des Computers großgeschrieben, Ihre Befehle dagegen fett gedruckt.

Verwenden Sie für die Anpassung eine Kopie Ihrer CP/M-Diskette, nicht das Original.

Durchführung der Anpassung

Sie benötigen zur Anpassung eine Pascal-Diskette. Sollten Sie kein Pascal-System haben, so bitten Sie Ihren Händler, für Sie die folgende Procedur auszuführen. Im folgenden ist die Version beschrieben, wenn Sie das System UCSD IV.0 verwenden. Haben Sie das System APPLE1 , so lassen Sie jeweils das .IV hinter dem ZAP fort.

1. Laden Sie nun als erstes Ihr Pascalsystem in Laufwerk 4 und dann die Diskette ZAP: in Laufwerk 5.

2. Starten Sie das Programm ZAP.IV auf der Diskette ZAP durch den Befehl X und antworten Sie entsprechend dem Fettdruck im folgenden:

```
             X
EXECUTE WHAT FILE ? ZAP:ZAP.IV (RETURN)
VERSION IV.O ZAP, 27-MAY-1982            (das datum muß nicht
(c) SANDOR SCARI 1982                     identisch sein.)
COMMAND CONSOLE: COMMAND 'ZAP:CPM'
```

ACHTUNG: bevor Sie Return drücken, müssen Sie nun Ihre CP/M-Diskette in Drive 4 stecken. Es findet keine Prüfung, ob die CP/M-Diskette wirklich in Laufwerk 4 steckt, statt.

(RETURN)

Erst nach dem Drücken der (RETURN)-Taste wird das CP/M-System angepaßt.

Eigenschaften der CP/M Diskette nach der Anpassung

Dem logischen Drucker LST: kann mit Hilfe des Stat-Programms

entweder PLT: (Parallelprinter)
oder UL1: (serieller Printer)

zugeordnet werden.

PUN: kann UP1:
und
RDR: kann UR1:

zugeordnet werden.

Die serielle Schnittstelle (UP1: und UL1:)
hat die voreingestellte Baudrate von 9600 Bits/s.
Übertragen werden: 8 Datenbits, 2 Stoppbits, kein Paritätsbit.

Die Baudrate kann durch Beschreiben der Adresse $F280 eingestellt werden, siehe nächste Seite.

Wie Sie den entsprechenden Handbücher über CP/M entnehmen können, haben Sie hier Änderungsmöglichkeiten über DDT.

Befehl in $F280	Baudrate
$91	50
$92	75
$93	110
$94	134,5
$95	150
$96	300
$97	600
$98	1200
$99	1800
$9A	2400
$9B	3600
$9C	4800
$9D	7200
$9E	9600
$9F	19200

Zu 3. **Anpassung des Applesoft oder Integer Basics von Apple**

Bevor Sie von der ZAP:-Diskette die gewünschte Basicversion laden können, müssen die Files INTBAS.DATA und FPBAS.DATA von der BASICS:-Diskette, die mit den Floppydisklaufwerken mitgeliefert wird, mit Hilfe des Pascalsystems auf die ZAP:-Diskette kopiert werden.
Stecken Sie zu diesem Zweck die Diskette UCSD IV.0 (oder APPLE1, dann entfällt jeweils das .IV in den Kommandos) in Laufwerk 1, die Diskette ZAP: in Laufwerk 2 und schalten Sie nun den Rechner ein.
Sollte zuvor die Modifizierung des Pascalsystems vorgenommen worden sein, so befinden sich auf der Zap:-Diskette noch die Files SYSTEM.APPLE und NEW.APPLE, die aus Platzgründen wieder gelöscht werden müssen.
Um ein File löschen zu können, muß die Taste R (für Remove) gedrückt werden.
Wie bislang werden Ihre Befehlseingaben fett gedruckt und die Ausgaben groß geschrieben:

F
FILER: G, S, N, L, R, C, T, D, Q, W, B, E, K, M, P, V, X, Z
R
REMOVE WHAT FILE ? **ZAP:=.APPLE** (RETURN)
ZAP:SYSTEM.APPLE - REMOVED
ZAP:NEW.APPLE - REMOVED

UPDATE DIRECTORY ? **Y**

Sollten beide Files schon nicht mehr auf der Diskette sein, so erscheint auf dem Bildschirm anstelle der Bestätigung die Meldung:

FILE NOT FOUND

Um den freien Speicherplatz auf der Diskette voll nutzen zu können ist es nötig, durch Drücken der Taste K die Crunch-Routine zu starten.

K
CRUNCH WHAT VOL ? **ZAP:** (RETURN)
FROM END OF DISK; BLOCK 280 ? (Y/N) **Y**
. . . .
ZAP: CRUNCHED ,

werden Files verschoben, so wird dies auf dem Bildschirm angezeigt . . . .
Tauschen Sie nun die Diskette APPLE 1: in Laufwerk 1 gegen die BASICS-Diskette aus.
Machen Sie weiter mit Drücken der Taste T (für Transfer):

T
TRANSFER WHAT FILE ? **BASICS:=BAS.DATA** (RETURN)
TO WHERE? **ZAP:$** (RETURN) .

In Laufwerk 1 muß nun die BASICS:-Diskette wieder gegen die APPLE1:-Diskette ausgetauscht werden. Drücken der Taste Q läßt wieder die Kommandozeile auf dem Bildschirm erscheinen.
Um aus den transferierten Files die verschiedene Basicversion zu erzeugen, muß das auf der ZAP:-Diskette befindliche Programm ZAP gestartet werden.
Drücken Sie zu diesem Zweck die Taste X (für Execute), zunächst jedoch:

Q

X
EXECUTE WHAT FILE ? **ZAP:ZAP.IV** (RETURN)
VERSION 2.0 ZAP, 29-MARCH-82 c(datum kann anders sein)
(C) SANDOR SCARI 1982
BUFFER SIZE: 56 BLOCKS

COMMAND 'CONSOLE:'
COMMAND 'ZAP: BASIC' (RETURN)

Nach Ablauf des Programms können die verschiedenen Basicversionen von der Zap:-Diskette geladen werden.

Laden des Basics

Da der BASIS 108 kein Basic in ROMs hat, muß bei Verwendung von Basicprogrammen nach dem Einschalten einmal die gewünschte Basicversion geladen werden.

Legen Sie die ZAP:-Diskette in Laufwerk 1 und schalten Sie den Rechner ein. Auf dem Bildschirm erscheint nun:

```
INTERPRETER FILES:          (die reihenfolge kann auch
A: FPBAS.DATA                vertauscht sein.)
B: INTBAS.DATA
C: VC.16
D: FP 40
E: FP 80
F: INT 40
```

Sie können nun die gewünschte Version mit einem der Buchstaben A ... F wählen.

! Sollte auf dem Bildschirm keine derartige Auflistung zu
! sehen sein, sind die am Anfang dieses Punktes beschriebenen
! Tätigkeiten noch nicht, oder nicht richtig ausgeführt worden.

Beschreibung der Basicversionen

FPBAS.DATA

Original Applesoft mit Apple-Autostart-Monitor (Der BASIS 108 verhält sich wie ein Apple II mit Applesoft).

INTBAS.DATA

Apple Integer Basic mit Apple-Autostart-Monitor (Der BASIS 108 verhält sich wie ein Apple II mit Integerbasic).

VC.16

Muß vorgeladen werden, bevor **Visicalc** geladen wird.

FP40

Floatingpointbasic mit 40-Zeichendarstellung, Cursorblock ist aktiv, Groß/Kleinschreibung, Funktionstasten liefern ASCII-Zeichen 128.

FP80

Floatingpointbasic mit 80-Zeichendarstellung,
Cursorblock ist aktiv, Groß/Kleinschreibung,
Funktionstasten liefern ASCII-Zeichen 128

INT40

Integerbasic mit 40-Zeichendarstellung,
Cursorblock ist aktiv, Groß/Kleinschreibung,
Funktionstasten liefern ASCII-Zeichen 128.

Die FP-Versionen sind verbessertes Applesoft, die Verbesserungen bzw. Zusatzmöglichkeiten entnehmen Sie bitte Anhang D.

Sie arbeiten nun mit der entsprechenden Version des Basics, die Sie gewählt haben, indem Sie die entsprechende DOS-System-Diskette in das Laufwerk 1 einlegen und (RETURN) drücken.

A N H A N G   B

Volume UT108:

Auf der Rückseite der ZAP:-Diskette befinden sich einige nützliche Programme, die unter den Betriebssystemen Pascal, CP/M und DOS eingesetzt werden können. Folgende Möglichkeiten sind gegeben:

> Anpassung an verschiedene Drucker,
> Veränderung des Bildschirm-Zeichensatzes,
> Erhöhung der Diskettenkapazität (nur unter Pascal),
> Serielle Schnittstelle und Kleinschreibung unter DOS,
> Demonstrations-Programme.

Benutzung der Diskette unter Apple Pascal Version 1.1

DISPLAY.TEXT und DISPLAY.CODE, DISPLAY.A2.TEXT und DISPLAY.A2.CODE

Stellen Sie zunächst fest, welche Revisionsnummer Ihr Computersystem hat. Für Systeme mit der Revisionsnummer A2, die vor Sommer 1982 ausgeliefert wurden, wählen Sie die Programme DISPLAY.A2.TEXT und DISPLAY.A2.CODE. Sie finden diese Nummer auf der Hauptplatine. Mit dem Programm DISPLAY.CODE lassen sich die verfügbaren Zeichensätze des BASIS 108 darstellen und durch die entsprechende Eingabe umstellen. Die Umstellung ist aber nur temporär und läßt sich mit diesem Programm nicht auf der Boot-Diskette festhalten. (Wenn Sie eine Änderung auf der Diskette vornehmen wollen, so können Sie dies mit dem Programm PRNT/V24.CODE erreichen.) DISPLAY.TEXT ist das dazugehörige Textfile.

X Execute what file? **UT108:DISPLAY** (RETURN)

FORMAT40.CODE

Mit diesem Programm können Sie die Speicherkapazität von 5 1/4" Disketten auf 160 KByte erhöhen, sofern Sie die entsprechenden Laufwerke besitzen. Dies geschieht durch Formattierung von 40 Spuren.

X Execute what file? **UT108:FORMAT40** (RETURN)

PRNT/V24.CODE

Mit diesem Programm können Sie den BASIS 108 an die Erfordernisse ihres Druckers anpassen. Dabei lassen sich folgende Parameter ändern:

```
Baudrate                         ( 50..19200 )
Databits                         ( 5,6,7,8 )
Parity                           ( j/n )
Stopbits                         ( 1,2 )
Printer: an V24-Schnittstelle    ( j/n )
Bildschirm-Zeichensatz
```

Die Änderung des Bildschirm-Zeichensatzes läßt sich auf der Bootdiskette eintragen, so daß der angewählte Zeichensatz beim erneuten Booten automatisch eingestellt wird.

X   Execute what file? **UT108:PRNT/V24**   **(RETURN)**

6551.TEXT

Dieses Textfile ist der modifizierte Treiber für die serielle Schnittstelle.

Benutzung der Diskette unter CP/M

DEUTSCH, ASCII, APL

Die auf der Diskette verfügbaren Files APL, ASCII, DEUTSCH ermöglichen eine Veränderung des Bildschirm-Zeichensatzes, die durch Aufruf des entsprechenden Programmes realisiert wird. Beispiel:

DEUTSCH (RETURN)

Hiermit stellen Sie den BASIS 108 auf den deutschen Zeichensatz um.

REBOOT

Wenn Sie dieses Programm ausführen, haben Sie die Möglichkeit, das System durch Eingabe von SHIFT SHIFT CONTROL von der Tastatur aus neu zu booten.

REBOOT (RETURN)

SYSWRT

Mit diesem Programm können Sie Boot-Disketten für den BASIS 108 herstellen. Die Disketten müssen formatiert sein.

SYSWRT (RETURN)

V24

Mit diesem Programm können Sie den BASIS 108 an die Erfordernisse ihres Druckers anpassen. Dabei lassen sich folgende Parameter ändern:

```
Baudrate                        ( 50..19200 )
Databits                        ( 5,6,7,8 )
Parity                          ( j/n )
Stopbits                        ( 1,2 )
Printer: an V24-Schnittstelle   ( j/n )
Bildschirm-Zeichensatz
```

Auch die Änderung des Bildschirm-Zeichensatzes läßt sich auf der Boot-Diskette eintragen, so daß der angewählte Zeichensatz beim erneuten Booten automatisch eingestellt wird.
Wichtig: Da beim erneuten Booten die V24-Schnittstelle nicht automatisch angesprochen wird, müssen Sie folgende Zuweisung unter CP/M tätigen.

STAT LST:=UL1:

Benutzung der Diskette unter DOS

Die deutsche Programmversion wird durch ein D hinter dem Programmnamen gekennzeichnet.

PRINTER/V24 und PRINTER/V24 D

PRINTER/V24 V2.1 und PRINTER/V24 V2.1 D

Mit diesen Programmen können Sie den BASIS 108 an die Erfordernisse ihres Druckers anpassen. Für die Anpassung brauchen Sie nur eines der Programme aufzurufen, die für Ihr Computersystem richtige Version wird automatisch ausgeführt. Es lassen sich folgende Parameter ändern:

```
Drucker und V24  CR- CR/LF Übersetzung ( n,j )
Drucker und V24  Bildschirmecho        ( n,j )
V24 Baudrate                           ( 50..19200 )
V24 Databits                           ( 5,6,7,8 )
V24 Paritätsbit                        ( j/n )
V24 Stopbits                           ( 1,2 )
```

DOS PATCH und DOS PATCH D

Nach der Ausführung dieses Programms läßt sich die Kleinschreibung auch für DOS-Kommandos verwenden. Außerdem kann Kleinschrift aus Textfiles gelesen werden. Unter PR 9 läßt sich die serielle Schnittstelle ansprechen.

RENUMBER UPDATE und CHAIN UPDATE

Wenn Sie eine überarbeitete Version des Programms RENUMBER erhalten wollen, gehen Sie am besten wie folgt vor:

Laden Sie das Programm RENUMBER UPDATE von der Diskette UT108:

LOAD RENUMBER UPDATE,S6,D1

Dann legen Sie eine nicht schreibgeschützte Diskette mit dem File RENUMBER in das Laufwerk D1 und starten das Programm RENUMBER UPDATE.

RUN

Wenn keine Fehlermeldungen erscheinen, war die Überarbeitung erfolgreich.
Die überarbeitete Version des Programms CHAIN erhalten sie in der gleichen Weise. Ersetzen Sie bei den oben angegebenen Befehlen RENUMBER durch CHAIN.

NEW FP DEMO , CHRGEN und COLOR DEMO108

Diese Programme werden als Demonstrationsbeispiele zum Bildschirm-Zeichensatz und zur Farbdarstellung mitgeliefert. Weiterhin sei daraufhingewiesen, daß das FP80 BASIC einige Vorteile gegenüber dem Applesoft enthält.

A N H A N G   C

BASIS 108 Monitor-ROM

Bildschirm:

	Apple	BASIS 108	
	24x40	24x40*	24x80
	ESC-	HOME	
	ESC-E	(Pfeil Ecke oben links)	
	ESC-F	(Pfeil Ecke oben rechts)	
	ESC-I	(Pfeil oben)	
	ESC-M	(Pfeil unten)	
	ESC-J	(Pfeil links)	
	ESC-K	(Pfeil rechts)	
	(Pfeil links)	(Pfeil Ecke unten links)	
	(Pfeil rechts)	(Pfeil Ecke unten rechts)	

*Monitor-ROM mit 40 Zeichen/Zeile
oder entsprechende Version aus ZAP.

Kassette:			
	xxxx.yyyy R	xxxx.yyyy R	--------
	xxxx.yyyy W	xxxx.yyyy W	--------
BASIC Kaltstart (nur ohne Disk):			
	CTRL-B	CTRL-B	CTRL-B
BASIC Warmstart:			
ohne Disk.	CTRL-C	Q	Q
mit Disk.	3D0G	Q	Q
LO-RES	40x40	40x40	40x80
	48x40	48x40	48x80
Disas	xxxx.yyyyL	xxxx.yyyyL	xxxx.yyyyL

	Apple	BASIS 108	
	24x40	24x40*	24x80
Eingabe-Vector:			
	nCTRL-K	nK	nK
Ausgabe-Vector:			
	nCTRL-P	nP	nP
6502-Register zeigen:			
	CTRL-E	?	?
User-Programm:			
	CTRL-Y	U	U
Eingabe:	nur Groß-buchstaben	Groß-/Kleinbuchstaben	
6502 Programm starten:	xxxxG	xxxxG	xxxxG
Move	xxxx<yyyy.zzzzM	(unverändert)	
Verify	xxxx<yyyy.zzzzV	(unverändert)	
Display	xxxx.yyyy	(unverändert, zeigt jedoch 16 Bytes/Zeile).	

A N H A N G   D

Hinweise zu Applesoft BASIC FP40 und FP80

1. Folgende Fehler wurden beseitigt:

   FOR I=S TO P ist nicht mehr FOR I=STOP
   Da hier Blanks beachtet werden, müssen Befehle wie COLOR=,
   TAB(ohne Blank vor dem Sonderzeichen geschrieben werden.

   TAB(..), SPC(..), HTAB,
   (bleibt immer im eingestellten Bildfenster).
   S. Applesoft Ref. Manual, Seite 129.

   LEFT$(A$,0) ergibt String der Länge 0 ohne Fehlermeldung
   RIGHT$(A$,0) entsprechend.

2. Erweiterungen

   Bei der Version 80 Zeichen/Zeile können im Grafik Modus LORES
   80x40 oder 80x48 Bildpunkte gesetzt werden.

   Der INPUT-Befehl kann kleine und große Buchstaben annehmen, allerdings keine Zusatztasten.

   Der GET-Befehl unterstützt auch die Zusatztasten:

   ```
   GET A$: IF ASC(A$) 127 THEN PRINT "Zusatztaste"; ASC(A$)-160:
   ```

   Schlüsselwörter und Variable dürfen kleingeschrieben werden.

   Es gibt drei Möglichkeiten auszugeben:

   normal,
   flash und
   inverse.

   Das bedeutet aber, daß 3 * 96 = 256 Zeichen belegt sind, deshalb gibt es nicht gleichzeitig INVERSE und FLASH.

A N H A N G   E

V24 Parameter

6551 Register in RAM:	DOS BASIC	UCSD II.1.1 (6502 Adressen)	UCSD IV.0	CP/M (Z-80 Adr.)
Baudrate, Wortlänge, Stopbits: 6551 Control Reg.	$06F9	$FFCE	$0271	$F280
Parität: RTS, DTR 6551 Command Reg.	$0779	$FFCF	$0270	$F281
Gerätename Eingabe:	IN#9 *)	remin: #7:	remin: #7:	UR1:
Ausgabe:	PR#9 *)	remout: #8:	remout: #8:	UL1: UP1:

Paralleler Druckerausgang:

Gerätename Ausgabe:	PR#1	printer: #6:	printer: #6:	LPT:

*) Bemerkung:
DOS 3.3 erlaubt IN # und PR # nur im Bereich 0..7. Damit IN#9 und PR#9 in DOS auch möglich sind, muß   POKE 41153,10   geändert werden..

Disketten, die mit geändertem DOS angelegt werden, erlauben IN#9 und PR#9 ohne weitere POKE-Befehle.

CR - CR/LF Übersetzung (gilt nur für DOS/BASIC)

Übersetzung.	keine	ein	keine	ein
Bildschschirmecho	kein	kein	ein	ein
$0679:	$00	$80	$40	$C0
$05F9:	$A5	$25	$E5	$65.

Wenn die 2 Bytes bei $0679 und $05F9 nicht zusammenpassen, werden alle Drucker und V24-Parameter auf die Standarteinstellung gesetzt:

Standartein-
stellung des V24: 9600 Baud, 2 Stopbits, keine Parität
CR - CR/LF Übersetzung ein, Bildschirmecho ein.

Paralleler Druk-
kerausgang: CR- CR/LF Übersetzung ein, Bildschirmecho ein.

A N H A N G   F

Anschluß eines Fernsehgerätes ohne Videoeingang

Besorgen Sie sich bei Ihrem BASIS Vertriebspartner einen UHF-Modulator, der das Video-Signal in ein HF-Signal umwandelt.
Bitte lesen Sie zunächst S. 8 "Öffnen des Systems" und dann auch entsprechend auf S. 10 "Hauptplatine".
Ziehen Sie den Stecker auf der linken oberen Seite der Platine Verbindungskabel zum Außenstecker für Video) und befestigen Sie das lose Kabel mit einem Klebstreifen an der Gehäuserückwand. Stecken Sie nun den entsprechenden Stecker des Modulators auf die Stiftleiste. Den Modulator befestigen Sie am besten ebenfalls mit Klebstreifen an der Rückwand. Das Anschlußkabel für das Fernsehgerät wird vom Modulator durch den Durchbruch auf der Gehäuserückseite nach außen geführt.
Auf Kanal 36 (beachten Sie aber bitte hierzu die Angaben beim Modulator) können Sie die Datenausgabe Ihres BASIS 108 empfangen. Bitte bedenken Sie aber, daß die Qualität der Zeichendarstellung durch den Umweg über den Modulator leidet und nicht mit einem guten Monitor vergleichbar ist.
Es sei nochmals darauf hingewiesen, daß ein normales Fernsehgerät mehr als 40 Zeichen/Zeile nicht sauber darstellen kann.
Sollten Sie großen Wert auf gute Farbausgabe legen, dann benötigen Sie einen hochauflösenden RGB-Monitor. Ihr BASIS-Vertriebspartner wird Sie auch in dieser Angelegenheit beraten.

A N H A N G   G

Arbeiten mit dem Kassettenrekorder

Schreiben eines Speicherbereichs auf Kassette

Dieses Monitor-Kommando kann nur ausgeführt werden, wenn der Monitor ROM in Ihren BASIS 108 mit 40 Zeichen/ Zeile arbeitet. D.h., Sie können hiermit arbeiten, wenn Sie FPBAS.DATA, INTBAS.DATA, FP40 oder INT40 geladen haben. Wollen Sie allerdings von der Kassette Basic laden, arbeiten Sie also ohne Diskettenlaufwerk, dann benötigen Sie den Monitor ROM für 40 Zeichen/ Zeile. Die Unterschiede der beiden Monitor ROMs sind in Anhang M aufgelistet.
Zwei spezielle Kommandos ermöglichen es Ihnen Speicherbereiche auf die Kassette Ihres Kassettenrekorders zu schreiben und bei späterem Gebrauch wieder einzulesen. Das erste dieser beiden Kommandos, das WRITE-Kommando, schreibt den Inhalt von einer oder bis zu 65536 Speicherstellen auf die Kassette.

Um einen solchen Speicherbereich auf Kassette zu schreiben, geben Sie dem Monitor die Anfangs- und Endadresse des Speicherbereichs, gefolgt von einem W (für WRITE=Schreiben) ein.

Um fehlerfrei aufnehmen zu können, muß der Kassettenrekorder auf "Aufnahme" stehen, bevor Sie (RETURN) nach Ihrer Eingabe tippen. Lassen Sie das Band ein paar Sekunden laufen, bevor Sie (RETURN) tippen. Der Monitor schreibt eine 10 Sekunden lange Vorinformation (HEADER) auf das Band und dann erst die Daten. Sobald der Vorgang beendet ist, meldet der Monitor sich mit einem Ton aus dem Lautsprecher und wartet auf weitere Anweisungen. Sie können dann das Band zurückspulen, es aus dem Rekorder nehmen und mit einer Inhaltsangabe versehen.

Beispiel:

```
*0.14(RETURN)
 0000: FF FF AD 30 C0 88 D0 04 C6 01 F0 08 CA D0 F6 A6
 0010: 00 4C 02 00 60
*0.14W                  c(kassettenrekorder auf aufnahme
                           schalten und zehn sekunden
                           laufen lassen)
          (RETURN)
*
```

Es dauert ca. 20 Sekunden (einschl. der 10 Sekunden für die Vorinformation), um die Werte von 4096 Speicherstellen auf Band zu schreiben. Dabei werden ca. 3000 Bit pro Sekunde übertragen. Wenn alle Daten übertragen sind, schreibt der Monitor noch einen zusätzlichen Wert auf das Band; die "Prüfsumme", die aus allen übertragenen Werten des Speicherbereichs gebildet wird. Das READ-Kommando (siehe unten)

benutzt diesen Wert, um Übertragungsfehler festzustellen. Die Prüfsumme ist anfangs $FF und wird durch Exclusive-OR von jedem Wert des übertragenen Bereichs verändert.

Lesen eines Speicherbereichs von der Kassette

Den mit Hilfe des WRITE-Kommandos auf Band geschriebenen Speicherbereich können Sie mit dem READ-Kommando (Lesen) R wieder in einen von Ihnen zu bestimmenden Bereich einlesen.

Geben Sie auch hier nicht sofort das (RETURN), sondern stellen Sie den Kassettenrekorder auf "Wiedergabe" und warten Sie, bis das Vorspannband durchgelaufen ist. Obwohl das WRITE-Kommando eine 10 Sekunden lange Vorinformation geschrieben hat, braucht das READ-Kommando nur drei Sekunden, um sich auf die Frequenz einzustellen. Sie sollten also ein paar Sekunden vergehen lassen, bis Sie die (RETURN)-Taste tippen.

Beispiel:

```
*0:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0(RETURN)
*0.14(RETURN)
 0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0010: 00 00 00 00 00
*0.14R                  c(kassettenrekorder einschalten
                            einige sekunden warten)
      (RETURN)
*0.14(RETURN)
 0000: FF FF AD 30 C0 88 D0 04 C6 01 F0 08 CA D0 F6 A6
 0010: 00 4C 02 00 60
*
```

Nachdem der Monitor alle Werte gelesen und gespeichert hat, liest er die auf Band gespeicherte Prüfsumme und vergleicht sie mit der soeben beim Lesen erstellten Prüfsumme. Weichen beide Werte voneinander ab, gibt der Monitor ein Signal zum Lautsprecher und schreibt ERR (Fehler) auf den Bildschirm. Sie erhalten also eine Warnung, daß beim Lesen der Daten ein Fehler aufgetreten ist und die im Speicher befindlichen Werte nicht mit den aufgezeichneten Werten übereinstimmen. Wenn die Prüfsumme stimmt, erwartet der Monitor weitere Anweisungen von Ihnen.

Wichtig

Es sei hier nochmals darauf hingewiesen, daß die soeben behandelten Kommandos W und R nur in dem Monitor ROM für 40 Zeichen/ Zeile vorhanden sind.
Siehe auch Anhang M.

A N H A N G   H

Hexadezimalzahlen

Eine Vielzahl von Adressen und Werten, vor allem im Monitor ROM oder bei Arbeiten mit anderen Speichern, benötigt man die Angaben in hexadezimaler Schreibweise.
Diese Schreibweise verwendet neben den Ziffern 0 bis 9 zusätzlich die Buchstaben A bis F, um die Werte 10 bis 15 darzustellen. Eine Hexadezimalziffer kann deshalb die Werte von 0 bis 15 annehmen. Damit stellen also zwei Hexadezimalziffern die Dezimalzahlen von 0 bis 255 und eine Gruppe von vier Ziffern den Bereich von 0 bis 65535 dar.
Eine Adresse wird im BASIS 108 also durch vier Hexadezimalziffern und jeder Wert (Inhalt einer Speicherstelle) durch zwei Hexadezimalziffern dargestellt. Um die Umrechnung Hexadezimalziffern in Dezimalzahlen zu erleichtern und zu veranschaulichen dient die folgende Tabelle.

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

A N H A N G   I

Tabelle der Tastenbelegung

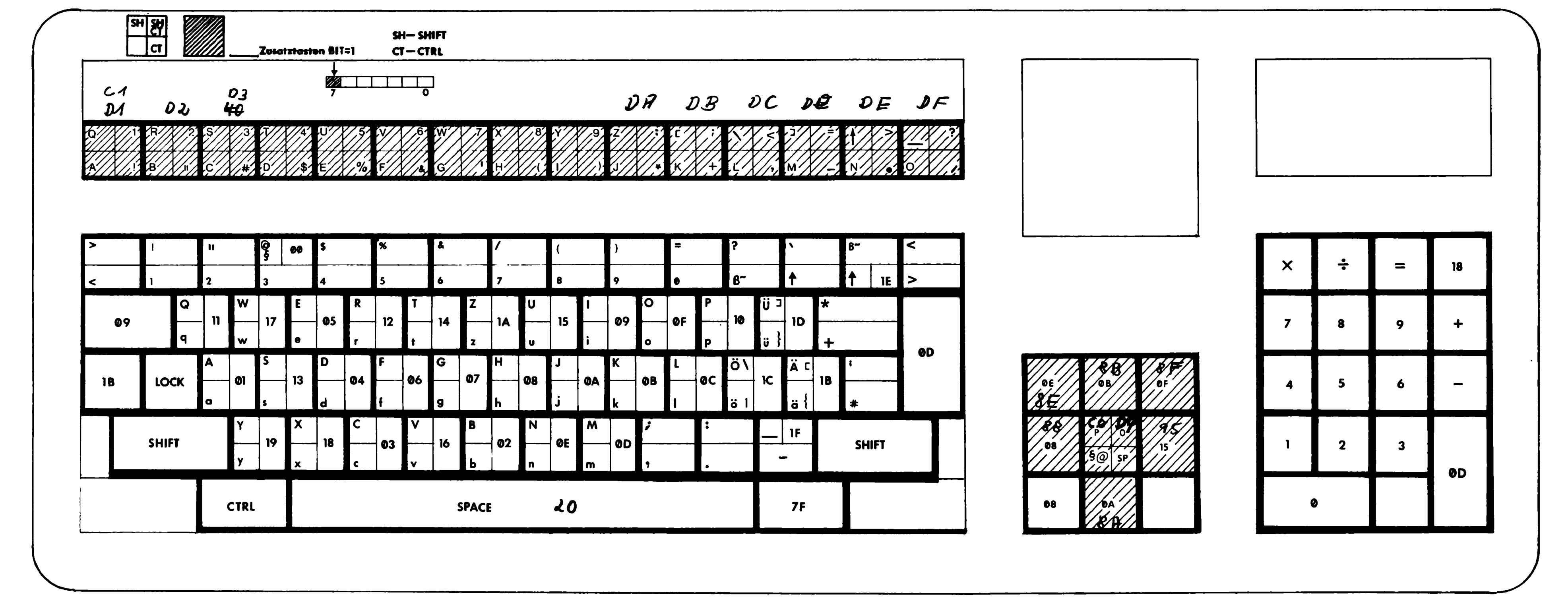
In der folgenden Tabelle wird der ASCII-Zeichensatz mit der Tastenbelegung und den zugehörigen Hexadezimalzahlen aufgeführt.
Da die Zifferntastatur nur immer entsprechend einfach belegt ist, wird hier nur das Haupttastenfeld und der Cursorblock behandelt.
Es gelten folgende Abkürzungen: CT - CTRL, SH - SHIFT.
Werden Zeichen bei den Tasten durch einen Bindestrich verbunden, so bedeutet das, daß diese Tasten gleichzeitig gedrückt werden müssen.

Hex.	ASCII	Taste	Hex.	ASCII	Taste
$00	nul	CT-SH-3	$20	space	Space
$01	soh	CT-a	$21	!	SH-1
$02	stx	CT-b	$22	"	SH-2
$03	etx	CT-c	$23	#	#
$04	eot	CT-d	$24	$	SH-4
$05	enq	CT-e	$25	%	SH-5
$06	ack	CT-f	$26	&	SH-6
$07	bel	CT-g	$27	'	SH-#
$08	bs	⇐	$28	(	SH-8
$09	ht	TAB	$29	)	SH-9
$0A	lf	CT-j	$2A	*	SH-+
$0B	vt	CT-k	$2B	+	+
$0C	ff	CT-l	$2C	,	,
$0D	cr	CT-m	$2D	-	-
$0E	so	CT-n	$2E	.	.
$0F	si	CT-o	$2F	/	SH-7
$10	dle	CT-p	$30	0	0
$11	dc1	CT-q	$31	1	1
$12	dc2	CT-r	$32	2	2
$13	dc3	CT-s	$33	3	3
$14	dc4	CT-t	$34	4	4
$15	nak	⇒	$35	5	5
$16	syn	CT-v	$36	6	6
$17	etb	CT-w	$37	7	7
$18	can	CT-x	$38	8	8
$19	em	CT-y	$39	9	9
$1A	sub	CT-z	$3A	:	SH-.
$1B	esc	ESC	$3B	;	SH-,
$1C	fs	CT-ö = |	$3C	< = μ	SH->
$1D	gs	CT-ü = }	$3D	=	SH-0
$1E	rs	CT-↑	$3E	> = °	>
$1F	us	CT-SH--	$3F	?	SH-ß

Hex.	ASCII	Taste
$40	§ =@	SH-3
$41	A	SH-a
$42	B	SH-b
$43	C	SH-c
$44	D	SH-d
$45	E	SH-e
$46	F	SH-f
$47	G	SH-g
$48	H	SH-h
$49	I	SH-i
$4A	J	SH-j
$4B	K	SH-k
$4C	L	SH-l
$4D	M	SH-m
$4E	N	SH-n
$4F	O	SH-o
$50	P	SH-p
$51	Q	SH-q
$52	R	SH-r
$53	S	SH-s
$54	T	SH-t
$55	U	SH-u
$56	V	SH-v
$57	W	SH-w
$58	X	SH-x
$59	Y	SH-y
$5A	Z	SH-z
$5B	Ä=[	SH-ä
$5C	Ö=\	SH-ö
$5D	Ü=]	SH-ü
$5E	^='	^
$5F	—	SH--

Hex.	ASCII	Taste
$60	`	SH-'
$61	a	a
$62	b	b
$63	c	c
$64	d	d
$65	e	e
$66	f	f
$67	g	g
$68	h	h
$69	i	i
$6A	j	j
$6B	k	k
$6C	l	l
$6D	m	m
$6E	n	n
$6F	o	o
$70	p	p
$71	q	q
$72	r	r
$73	s	s
$74	t	t
$75	u	u
$76	v	v
$77	w	w
$78	x	x
$79	y	y
$7A	z	z
$7B	ä={	ä
$7C	ö=|	ö
$7D	ü=}	ü
$7E	ß=@	ß
$7F	del	DELETE

Da es die ASCII-Zeichen in unterschiedlichen Versionen (z.B. US- oder deutsch) gibt und die Tastatur diese Zeichen widergibt, kommen manche Zeichen mehrfach vor (z.B.   und ß) bzw. unterschiedliche Belegung (z.B. ö und).
Die Bedeutung der Cursorblocktasten können Sie entsprechend den Eintragungen im Tastenfeld entnehmen. Siehe nächste Seite.
Die Zusatztasten gehen mit Ihren Zeichen, die in dem Tastaturschema eingetragen sind, über den üblichen ASCII-Zeichensatz hinaus. Diese Zeichen sind aber im Vergleich zum normalen ASCII-Zeichensatz um 128 nach oben verschoben, d.h. Bit 7 ist 1 bei den ASCII-Werten dieser Tasten.


SH—SHIFT
CT—CTRL
Zusatztasten BIT=1
SHIFT
LOCK
CTRL
SPACE
SHIFT

A N H A N G J

Zusammenstellung der Ein-/Ausgabeadressen

Adresse	Lesen	Schreiben
$C000	Tastatur	Inverse
$C001		Flash
$C002		SW1 aus
$C003		SW1 ein
$C004		SW2 aus
$C005		SW2 ein
$C006		2 x 128 Zeichen
$C007		2 x 64 + 128 Zeichen
$C008	Tastaturerweiterung	Tastaturunterbrechung aus
$C009		Tastaturunterbrechung ein
$C00A		40 Zeichen/Zeile
$C00B		80 Zeichen/Zeile
$C00C		Statik RAM aus
$C00D		Statik RAM ein
$C00E		$C08x aktiv
$C00F		$C08x blockiert
$C010	Tastaturstrobe	
$C020	Kasettenausgang	
$C030	Lautsprecher	
$C04x	Utility Strobe	Utility Strobe
$C050	Graphik ein	
$C051	Graphik aus	
$C052	Vollgraphik	
$C053	mixed Graphik	
$C054	Seite 1 aktiv	
$C055	Seite 2 aktiv	
$C056	LO-RES-Graphik	
$C057	HI-RES-Graphik	
$C058	TTL-0 low	
$C059	TTL-0 high	
$C05A	TTL-1 low	
$C05B	TTL-1 high	
$C05C	TTL-2 low	
$C05D	TTL-2 high	
$C05E	TTL-3 low	
$C05F	TTL-3 high	

Adresse	Lesen	Schreiben
$C060	Kassette Eingang	$0000 - $1FFF Bank 0
$C061	TTL-Eingang 1	$0000 - $1FFF Bank 1
$C062	TTL-Eingang 2	$2000 - $3FFF Bank 0
$C063	TTL-Eingang 3	$2000 - $3FFF Bank 1
$C064	Handregler 0	$4000 - $5FFF Bank 0
$C065	Handregler 1	$4000 - $5FFF Bank 1
$C066	Handregler 2	$6000 - $7FFF Bank 0
$C067	Handregler 3	$6000 - $7FFF Bank 1
$C068		$8000 - $9FFF Bank 0
$C069		$8000 - $9FFF Bank 1
$C06A		$A000 - $BFFF Bank 0
$C06B		$A000 - $BFFF Bank 1
$C06C		$D000 - $DFFF Bank 0
$C06D		$D000 - $DFFF Bank 1
$C06E		$E000 - $FFFF Bank 0
$C06F		$E000 - $FFFF Bank 1
$C070	Handreglerstrobe	
$C08x	LC-Steuerung	
$C090		Drucker parallel Ausgang
$C098	seriell Eingang	seriell Ausgang
$C099	seriell Status	seriell RESET
$C09A	seriell Command	seriell Command
$C09B	seriell Control	seriell Control
$C0Ax	Slot 2 DEVICE Select	Slot 2 DEVICE Select
.	.	.
$C0Fx	Slot 7 DEVICE Select	Slot 7 DEVICE Select
$C100		Z80 ein/aus
$C1C1	Drucker Acknowledge	

A N H A N G   K

Der Z-80-Teil

Einleitung

Der Z-80-Teil beinhaltet die notwendige Hardware, um einen Z-80 Mikroprozessor an den BUS anzupassen. Dadurch ist die direkte Ausführung des 8080 und Z-80 Programms einschließlich des CP/M-Betriebssystems möglich.
In das System ist die Language Card für das 56k CP/M oder ein anderes Programm, das unter CP/M arbeitet, integriert.

Taktgenerierung

Der Z-80 Mikroprozessor ist synchronisiert und mit dem 6502 Takt phasengekoppelt.
Während jeder Video Refresh Periode Φ_1 , wird der 7 Mhz Takt unterteilt, um 3 halbe Perioden von 135 ns zu ermöglichen.
Der erste halbe Takt ist immer höher, der zweite immer niedriger und der dritte wieder hoch.
Nach dem Ende des dritten halben Taktes geht das Signal auf logisch 0 und bleibt dort bis zum Start des nächsten Φ_1. Das bedeutet, daß der Z-80 Takt während des Systemtaktes Φ_0 und einem geringen Teil von Φ_1 logisch 0 ist. Der vierte Halbtakt ist 563 ns lang. (Diese Zeit wird um 69 ns am Ende eines jeden Videolaufes verlängert). Der effektive Z-80 Takt ist 2.041 MHz.
Jede Art von Maschinentakt beinhaltet eine Speicherzugangszeit Φ_0. Das Lese-/Schreibsignal wird durch Synchronisieren der ansteigenden Flanke des Schreibübergangs zum Z-80-Teil-Takt erzeugt und garantiert, daß das Schreiben während dieser Zeit nach logisch 0 geht und der Z-80-Teil nach logisch 1.
Da alle Adressübergänge vom Z-80 ausgehen, wenn deren Takte logisch 1 sind, müssen sie alle während Φ_1 mit den Videoerneuerungszugriffen erscheinen. Deswegen haben alle Φ_0 Takte feste Adressen für die ganze Dauer des Taktes.

Kontrolle des Z-80-Teiles

Der Z-80-Teil wird durch Schreibkommandos in den Speicherraum, der normalerweise periphere ROMs beinhaltet, kontrolliert. Es ist sehr wichtig, mit Schreibbefehlen zu arbeiten, um sicherzustellen, daß der 6502 nicht 2 Zugriffe hintereinander ausführt (dieses würde ein Zurückschalten auf den 6502 verhindern).
Wenn der BASIS 108 eingeschaltet ist, schaltet das (RESET)-Signal den Z-80-Teil aus. Das (RESET)-Signal ist mit dem internen Takt synchronisiert, um sicherzustellen, daß eine Schreiboperation nicht unterbrochen werden kann. Der Z-80 geht sofort in einen Wartemodus über und bleibt dort bis der Z-80-Teil aktiviert wird.
Nach Empfang eines Schreibbefehles im richtigen Speicherbereich ist der Z-80-Teil eingeschaltet. Der Z-80 bleibt in einem Wartemodus bis ein Speichertakt mit Adressinformationen für den Z-80-Teil erscheint. Jetzt wird der Z-80 vom

Wartemodus befreit und läuft nun ohne weitere Wartetakte.
Mit Empfang eines anderen Schreibbefehles im gleichen Speicherbereich (dieses Mal aus dem Z-80-Teil selbst) wird der Z-80-Teil ausgeschaltet.
Die Speicheradressen für die Kontrolle des Z-80-Teiles sind:

$C100 - $C1FF.

Anpassung des Adress Bus

Der Adress Bus des Z-80-Teiles ist an den BASIS 108 I/O Bus durch eine Adressübersetzung angepaßt. Diese Übersetzung beseitigt die Speicherprobleme, die zwischen der 6502 Architektur und den CP/M- und Z-80-Konventionen bestehen. Diese Logik addiert $1000 in allen Adressen, wenn er eingeschaltet ist. Der Dip-Schalter S1-1 ist dann aus. Dies verschiebt die Z-80 Interrupt-Adressen und die CP/M Startadressen aus der 0 Bank des 6502-Speichers.
Zusätzlich werden Adressen in den $C000-$EFFF-Bereichen verschoben, um dem CP/M angrenzende Speicher zu öffnen.
Die aufgeführte Tabelle zeigt genau, wie der Übersetzer funktioniert:

Z-80 Adressen	6502 Adressen
$0000-$0FFF	$1000-$1FFF
$1000-$1FFF	$2000-$2FFF
.	.
$A000-$AFFF	$B000-$BFFF
$B000-$BFFF	$D000-$DFFF
$C000-$CFFF	$E000-$EFFF
$D000-$DFFF	$F000-$FFFF
$E000-$EFFF	$C000-$CFFF
$F000-$FFFF	$0000-$0FFF.

Der Z-80 kann zusammenhängende Speicher von $0000-$DFFF adressieren, ohne die 0 Page des 6502 Prozessors und den I/O Bereich zu stören.
Wenn der Übersetzer ausgeschaltet ist (S1-1 eingeschaltet) erscheinen die gepufferten Z-80 Adressen unverändert auf dem I/O Bus.
Alle Puffer sind immer im hochohmigen Zustand, wenn der Z-80-Teil die Kontrolle über den Bus aufgibt. Die Zeitsteuerung beim Ein- und Ausschalten soll den Z-80-Teil daran hindern, auf den Adressenbus zuzugreifen, wenn andere Elemente die Bus-Kontrolle übernommen haben.
Die Zeitsteuerung des Z-80-Teiles zwingt alle Adressübergänge während der Zeit zu erscheinen, in der der Bildschirm durch den BASIS 108 aufgefrischt wird. Da für jeden Speicherzugriff die Adressen bereits bei Beginn des Zyklus stabil sind, ist kein Wartezyklus erforderlich.

DMA Daisy Chain

Der DMA Daisy Chain wird so lange durchgeführt, bis eine höher priviligierte DMA Device die Übernahme der Kontrolle des Bus vom Z-80 anfordert. Der eingeschaltete Dip-Schalter S1-2 ermöglicht es dem DMA, den Z-80-Teil zu unterbrechen. Wenn dieser Schalter eingeschaltet ist und die DMA Daisy Chain Leitung (Pin 24) nach 0 geht, wird der laufende Z-80 Maschinenzyklus beendet. Der

Z-80 zeigt die Freigabe des Bus durch die DMA-Leitung an. DMA geht auf logisch 0.
Zu diesem Zeitpunkt kann ein anderes Gerät die Kontrolle übernehmen, indem die DMA-Leitung logisch 0 gesetzt wird. Die Kontrolle darf durch das andere Gerät nicht früher übernommen werden, da bis zu diesem Zeitpunkt der Z-80 den Bus immer noch kontrolliert.

Der Z-80 hat die niedrigste DMA-Priorität.

Interrupts

Damit sowohl der Z-80 als auch der 6502 Mikroprozessor Interrupts erkennen können, wurde entsprechende Hardware integriert. Wenn der Dip-Schalter S1-4 eingeschaltet ist, erkennt der Z-80 Interrupts. Das Interruptprogramm sollte die Kontrolle an den 6502 für den weiteren Betrieb zurückgeben. So hat der 6502, der auch den Interrupt feststellte, die Möglichkeit sich vom Interruptstatus zu befreien.
Der Z-80 wird im Interruptmode 1 betrieben.
Der Dip-Schalter S1-3 hat die gleichen Funktionen für den nicht maskierbaren Interrupt.

Z8400 Z80® CPU Central Processing Unit



Product Specification

March 1981

Features

- The instruction set contains 158 instructions. The 78 instructions of the 8080A are included as a subset; 8080A software compatibility is maintained.
- Six MHz, 4 MHz and 2.5 MHz clocks for the Z80B, Z80A, and Z80 CPU result in rapid instruction execution with consequent high data throughput.
- The extensive instruction set includes string, bit, byte, and word operations. Block searches and block transfers together with indexed and relative addressing result in the most powerful data handling capabilities in the microcomputer industry.
- The Z80 microprocessors and associated family of peripheral controllers are linked by a vectored interrupt system. This system may be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining.
- Duplicate sets of both general-purpose and flag registers are provided, easing the design and operation of system software through single-context switching, background-foreground programming, and single-level interrupt processing. In addition, two 16-bit index registers facilitate program processing of tables and arrays.
- There are three modes of high speed interrupt processing: 8080 compatible, non-Z80 peripheral device, and Z80 Family peripheral with or without daisy chain.
- On-chip dynamic memory refresh counter.

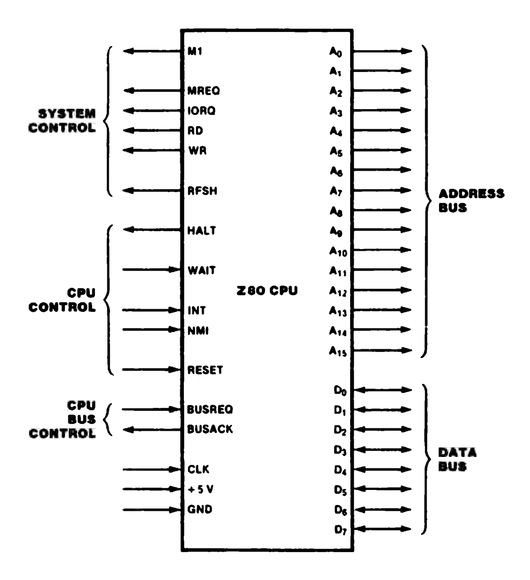


Figure 1. Pin Functions

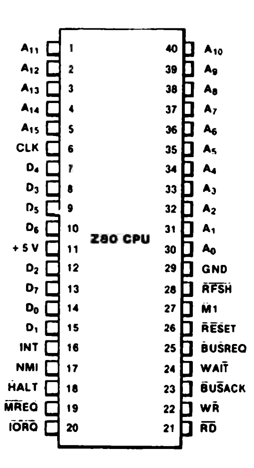


Figure 2. Pin Assignments

General Description

The Z80, Z80A, and Z80B CPUs are third-generation single-chip microprocessors with exceptional computational power. They offer higher system throughput and more efficient memory utilization than comparable second- and third-generation microprocessors. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general-purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers. A group of "Exchange" instructions makes either set of main or alternate registers accessible to the programmer. The alternate set allows operation in foreground-background mode or it may be reserved for very fast interrupt response.

The Z80 also contains a Stack Pointer, Program Counter, two index registers, a Refresh register (counter), and an Interrupt register. The CPU is easy to incorporate into a system since it requires only a single +5 V power source, all output signals are fully decoded and timed to control standard memory or peripheral circuits, and is supported by an extensive family of peripheral controllers. The internal block diagram (Figure 3) shows the primary functions of the Z80 processors. Subsequent text provides more detail on the Z80 I/O controller family, registers, instruction set, interrupts and daisy chaining, and CPU timing.

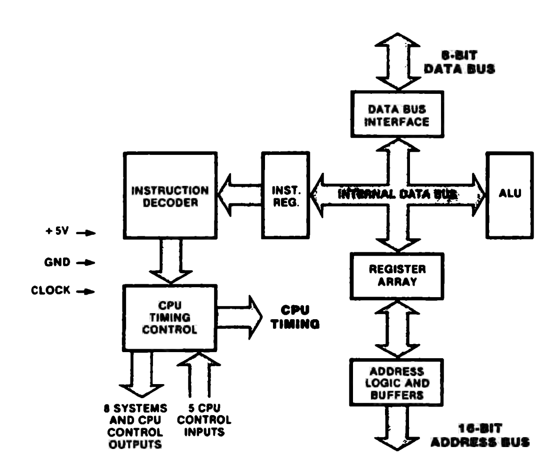


Figure 3. Z80 CPU Block Diagram

Z80 Microprocessor Family

The Zilog Z80 microprocessor is the central element of a comprehensive microprocessor product family. This family works together in most applications with minimum requirements for additional logic, facilitating the design of efficient and cost-effective microcomputer-based systems.

Zilog has designed five components to provide extensive support for the Z80 microprocessor. These are:

- The PIO (Parallel Input/Output) operates in both data-byte I/O transfer mode (with handshaking) and in bit mode (without handshaking). The PIO may be configured to interface with standard parallel peripheral devices such as printers, tape punches, and keyboards.
- The CTC (Counter/Timer Circuit) features four programmable 8-bit counter/timers, each of which has an 8-bit prescaler. Each of the four channels may be configured to operate in either counter or timer mode.
- The DMA (Direct Memory Access) controller provides dual port data transfer operations and the ability to terminate data transfer as a result of a pattern match.
- The SIO (Serial Input/Output) controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication, including Bi-Synch and SDLC.
- The DART (Dual Asynchronous Receiver/Transmitter) device provides low cost asynchronous serial communication. It has two channels and a full modem control interface.

Z80 CPU Registers

Figure 4 shows three groups of registers within the Z80 CPU. The first group consists of duplicate sets of 8-bit registers: a principal set and an alternate set (designated by ' [prime], e.g., A'). Both sets consist of the Accumulator Register, the Flag Register, and six general purpose registers. Transfer of data between these duplicate sets of registers is accomplished by use of "Exchange" instructions. The result is faster response to interrupts and easy, efficient implementation of such versatile programming techniques as background-foreground data processing. The second set of registers consists of six registers with assigned functions. These are the I (Interrupt Register), the R (Refresh Register), the IX and IY (Index Registers), the SP (Stack Pointer), and the PC (Program Counter). The third group consists of two interrupt status flip-flops, plus an additional pair of flip-flops which assists in identifying the interrupt mode at any particular time. Table 1 provides further information on these registers.

MAIN REGISTER SET		ALTERNATE REGISTER SET	
A ACCUMULATOR	F FLAG REGISTER	A' ACCUMULATOR	F' FLAG REGISTER
B GENERAL PURPOSE	C GENERAL PURPOSE	B' GENERAL PURPOSE	C' GENERAL PURPOSE
D GENERAL PURPOSE	E GENERAL PURPOSE	D' GENERAL PURPOSE	E' GENERAL PURPOSE
H GENERAL PURPOSE	L GENERAL PURPOSE	H' GENERAL PURPOSE	L' GENERAL PURPOSE

← 8 BITS →

← 16 BITS →

IX INDEX REGISTER	
IY INDEX REGISTER	
SP STACK POINTER	
PC PROGRAM COUNTER	
I INTERRUPT VECTOR	R MEMORY REFRESH

← 8 BITS →

INTERRUPT FLIP-FLOPS STATUS

IFF1 IFF2

0 = INTERRUPTS DISABLED
1 = INTERRUPTS ENABLED

STORES IFF1 DURING NMI SERVICE

INTERRUPT MODE FLIP FLOPS

IMF_a	IMF_b	
0	0	INTERRUPT MODE 0
0	1	NOT USED
1	0	INTERRUPT MODE 1
1	1	INTERRUPT MODE 2

Figure 4. CPU Registers

Z80 CPU Registers (Continued)

Register		Size (Bits)	Remarks
A, A'	Accumulator	8	Stores an operand or the results of an operation.
F, F'	Flags	8	See Instruction Set.
B, B'	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C'	General Purpose	8	See B, above.
D, D'	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E'	General Purpose	8	See D, above.
H, H'	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L'	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte C — Low byte D — High byte E — Low byte H — High byte L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Automatically incremented and placed on the address bus during each instruction fetch cycle.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Same as IX, above.
SP	Stack Pointer	16	Stores addresses or data temporarily. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF_1-IFF_2	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

Table 1. Z80 CPU Registers

Interrupts: General Operation

The CPU accepts two interrupt input signals: $\overline{\text{NMI}}$ and $\overline{\text{INT}}$. The $\overline{\text{NMI}}$ is a non-maskable interrupt and has the highest priority. $\overline{\text{INT}}$ is a lower priority interrupt since it requires that interrupts be enabled in software in order to operate. Either $\overline{\text{NMI}}$ or $\overline{\text{INT}}$ can be connected to multiple peripheral devices in a wired-OR configuration.

The Z80 has a single response mode for interrupt service for the non-maskable interrupt. The maskable interrupt, $\overline{\text{INT}}$, has three programmable response modes available. These are:

- Mode 0 — compatible with the 8080 microprocessor.
- Mode 1 — Peripheral Interrupt service, for use with non-8080/Z80 systems.
- Mode 2 — a vectored interrupt scheme, usually daisy-chained, for use with Z80 Family and compatible peripheral devices.

The CPU services interrupts by sampling the $\overline{\text{NMI}}$ and $\overline{\text{INT}}$ signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected. Details on interrupt responses are shown in the CPU Timing Section.

Interrupts: General Operation (Continued)

Non-Maskable Interrupt ($\overline{NMI}$). The non-maskable interrupt cannot be disabled by program control and therefore will be accepted at at all times by the CPU. $\overline{NMI}$ is usually reserved for servicing only the highest priority type interrupts, such as that for orderly shutdown after power failure has been detected. After recognition of the $\overline{NMI}$ signal (providing $\overline{BUSREQ}$ is not active), the CPU jumps to restart location 0066H. Normally, software starting at this address contains the interrupt service routine.

Maskable Interrupt ($\overline{INT}$). Regardless of the interrupt mode set by the user, the Z80 response to a maskable interrupt input follows a common timing cycle. After the interrupt has been detected by the CPU (provided that interrupts are enabled and $\overline{BUSREQ}$ is not active) a special interrupt processing cycle begins. This is a special fetch ($\overline{M1}$) cycle in which $\overline{IORQ}$ becomes active rather than $\overline{MREQ}$, as in a normal $\overline{M1}$ cycle. In addition, this special $\overline{M1}$ cycle is automatically extended by two $\overline{WAIT}$ states, to allow for the time required to acknowledge the interrupt request and to place the interrupt vector on the bus.

Mode 0 Interrupt Operation. This mode is compatible with the 8080 microprocessor interrupt service procedures. The interrupting device places an instruction on the data bus, which is then acted on six times by the CPU. This is normally a Restart Instruction, which will initiate an unconditional jump to the selected one of eight restart locations in page zero of memory.

Mode 1 Interrupt Operation. Mode 1 operation is very similar to that for the $\overline{NMI}$. The principal difference is that the Mode 1 interrupt has a vector address of 0038H only.

Mode 2 Interrupt Operation. This interrupt mode has been designed to utilize most effectively the capabilities of the Z80 microprocessor and its associated peripheral family. The interrupting peripheral device selects the starting address of the interrupt service routine. It does this by placing an 8-bit address vector on the data bus during the interrupt acknowledge cycle. The high-order byte of the interrupt service routine address is supplied by the I (Interrupt) register. This flexibility in selecting the interrupt service routine address allows the peripheral device to use several different types of service routines. These routines may be located at any available location in memory. Since the interrupting device supplies the low-order byte of the 2-byte vector, bit 0 (A_0) must be a zero.

Interrupt Priority (Daisy Chaining and Nested Interrupts). The interrupt priority of each peripheral device is determined by its physical location within a daisy-chain configuration. Each device in the chain has an interrupt enable input line (IEI) and an interrupt enable output line (IEO), which is fed to the next lower priority device. The first device in the daisy chain has its IEI input hardwared to a High level. The first device has highest priority, while each succeeding device has a corresponding lower priority. This arrangement permits the CPU to select the highest priority interrupt from several simultaneously interrupting peripherals.

The interrupting device disables its IEO line to the next lower priority peripheral until it has been serviced. After servicing, its IEO line is raised, allowing lower priority peripherals to demand interrupt servicing.

The Z80 CPU will nest (queue) any pending interrupts or interrupts received while a selected peripheral is being serviced.

Interrupt Enable/Disable Operation. Two flip-flops, IFF_1 and IFF_2, referred to in the register description are used to signal the CPU interrupt status. Operation of the two flip-flops is described in Table 2. For more details, refer to the *Z80 CPU Technical Manual* and *Z80 Assembly Language Manual.*

Action	IFF_1	IFF_2	Comments
CPU Reset	0	0	Maskable interrupt $\overline{INT}$ disabled
DI instruction execution	0	0	Maskable interrupt $\overline{INT}$ disabled
EI instruction execution	1	1	Maskable interrupt $\overline{INT}$ enabled
LD A,I instruction execution			IFF_2 → Parity flag
LD A,R instruction execution			IFF_2 → Parity flag
Accept $\overline{NMI}$	0	IFF_1	IFF_1 → IFF_2 (Maskable interrupt $\overline{INT}$ disabled)
RETN instruction execution	IFF_2		IFF_2 → IFF_1 at completion of an $\overline{NMI}$ service routine.

Table 2. State of Flip-Flops

Instruction Set

The Z80 microprocessor has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor. It includes such unique operations as a block move for fast, efficient data transfers within memory or between memory and I/O. It also allows operations on any bit in any location in memory.

The following is a summary of the Z80 instruction set and shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. The *Z80 CPU Technical Manual* (03-0029-01) and *Assembly Language Programming Manual* (03-0002-01) contain significantly more details for programming use.

The instructions are divided into the following categories:

- □ 8-bit loads
- □ 16-bit loads
- □ Exchanges, block transfers, and searches
- □ 8-bit arithmetic and logic operations
- □ General-purpose arithmetic and CPU control
- □ 16-bit arithmetic operations
- □ Rotates and shifts
- □ Bit set, reset, and test operations
- □ Jumps
- □ Calls, returns, and restarts
- □ Input and output operations

A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:

- □ Immediate
- □ Immediate extended
- □ Modified page zero
- □ Relative
- □ Extended
- □ Indexed
- □ Register
- □ Register indirect
- □ Implied
- □ Bit

8-Bit Load Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r, r'	r ← r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	r, r' Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110 ← n →		2	2	7	
LD r, (HL)	r ← (HL)			X		X	•			01 r 110		1	2	7	
LD r, (IX+d)	r ← (IX+d)			X		X	•			11 011 101 01 r 101 ← d →	DD	3	5	19	
LD r, (IY+d)	r ← (IY+d)			X		X				11 111 101 01 r 110 ← d →	FD	3	5	19	
LD (HL), r	(HL) ← r			X		X				01 110 r		1	2	7	
LD (IX+d), r	(IX+d) ← r			X		X				11 011 101 01 110 r ← d →	DD	3	5	19	
LD (IY+d), r	(IY+d) ← r			X	•	X				11 111 101 01 110 r ← d →	FD	3	5	19	
LD (HL), n	(HL) ←			X		X	•			00 110 110 ← n →	36		3	10	
LD (IX+d), n	(IX+d) ← n			X	•	X	•			11 011 101 00 110 110 ← d → ← n →	DD 36		5	19	
LD (IY+d), n	(IY+d) ← n			X	•	X				11 111 101 00 110 110 ← d → ← n →	FD 36		5	19	
LD A, (BC)	A ← (BC)			X	•	X	•			00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)			X	•	X	•			00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)			X	•	X	•			00 111 010 ← n → ← n →	3A	3	4	13	
LD (BC), A	(BC) ← A			X	•	X	•			00 000 010	02	:	2	7	
LD (DE), A	(DE) ← A			X	•	X	•			00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A			X	•	X	•			00 110 010 ← n → ← n →	32	3	4	13	
LD A, I	A ← I			X	0	X	IFF	0		11 101 101 01 010 111	ED 57	2	2	9	
LD A, R	A ← R			X	0	X	IFF	0		11 101 101 01 011 111	ED 5F	2	2	9	
LD I, A	I ← A			X	•	X	•			11 101 101 01 000 111	ED 47	2	2	9	
LD R, A	R ← A			X	•	X	•			11 101 101 01 001 111	ED 4F	2	2	9	

NOTES: r, r' means any of the registers A, B, C, D, E, H, L.
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag.
For an explanation of flag notation and symbols for mnemonic tables, see Symbolic Notation section following tables.

16-Bit Load Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001 ← n → ← n →	3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn			X		X				11 011 101 DD 00 100 001 21 ← n → ← n →			14	
LD IY, nn	IY ← nn			X		X				11 111 101 FD 00 100 001 21 ← n → ← n →			14	
LD HL, (nn)	H ← (nn + 1) L ← (nn)			X		X				00 101 010 2A ← n → ← n →			16	
LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)			X		X				11 101 101 ED 01 dd1 011 ← n → ← n →			20	
LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)			X		X				11 011 101 DD 00 101 010 2A ← n → ← n →		6	20	
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)			X		X				11 111 101 FD 00 101 010 2A ← n → ← n →			20	
LD (nn), HL	(nn + 1) ← H (nn) ← L			X		X				00 100 010 22 ← n → ← n →		5	16	
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L			X		X				11 101 101 ED 01 dd0 011 ← n → ← n →			20	
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L			X		X				11 011 101 DD 00 100 010 22 ← n → ← n →				
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L			X		X				11 111 101 FD 00 100 010 22 ← n → ← n →		6	20	
LD SP, HL	SP ← HL			X		X				11 111 001 F9		1	6	
LD SP, IX	SP ← IX			X		X				11 011 101 DD 11 111 001 F9		2	10	
LD SP, IY	SP ← IY			X		X				11 111 101 FD 11 111 001 F9				
PUSH qq	(SP − 2) ← qq_L (SP − 1) ← qq_H SP → SP − 2			X		X				11 qq0 101			11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP − 2) ← IX_L (SP − 1) ← IX_H SP → SP − 2			X		X				11 011 101 DD 11 100 101 E5	2			
PUSH IY	(SP − 2) ← IY_L (SP − 1) ← IY_H SP → SP − 2			X		X				11 111 101 FD 11 100 101 E5				
POP qq	qq_H ← (SP + 1) qq_L ← (SP) SP → SP + 2			X		X				11 qq0 001	1			
POP IX	IX_H ← (SP + 1) IX_L ← (SP) SP → SP + 2			X		X				11 011 101 DD 11 100 001 E1	2			
POP IY	IY_H ← (SP + 1) IY_L ← (SP) SP → SP + 2			X		X				11 111 101 FD 11 100 001 E1	2			

NOTES: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and l...
e.g. BC_L = C, AF_H = A

Exchange, Block Transfer, Block Search Groups

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE ↔ HL	•	•	X	•	X			•	11 101 011 EB	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	X	•	X			•	00 001 000 08		1	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	•	•	X	•	X			•	11 011 001 D9		1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP + 1) L ↔ (SP)			X		X				11 100 011 E3			19	
EX (SP), IX	IX_H ↔ (SP + 1) IX_L ↔ (SP)			X		X				11 011 101 DD 11 100 011 E3		6	23	
EX (SP), IY	IY_H ↔ (SP + 1) IY_L ↔ (SP)			X	•	X	•			11 111 101 FD 11 100 011 E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1			X	0	X	① 1	0		11 101 101 ED 10 100 000 A0	2		16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1 Repeat until BC = 0			X	0	X				11 101 101 ED 10 110 000 B0				If BC ≠ 0 If BC = 0

NOTE ① P/V flag is 0 if the result of BC ... 0 ... otherwise P/V ...

Exchange, Block Transfer, Block Search Groups (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LDD	(DE) ← (HL) DE ← DE – 1 HL ← HL – 1 BC ← BC – 1			X		X	① ↕				ED A8				
LDDR	(DE) ← (HL) DE ← DE – 1 HL ← HL – 1 BC ← BC – 1 Repeat until BC = 0			X		X				10 1 101 10 111 000	ED B8				If BC ≠ 0 If BC = 0
CPI	A – (HL) HL ← HL + 1 BC ← BC – 1		② ↕	X		X	① ↕			101 001	ED A1				
CPIR	A – (HL) HL ← HL + 1 BC ← BC – 1 Repeat until A = (HL) BC = 0		② ↕	X		X	① ↕			01	ED B1				If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A – (HL) HL ← HL – 1 BC ← BC – 1		② ↕	X		X	① ↕			11 101 101 10 101 001	ED A9				
CPDR	A – (HL) HL ← HL – 1 BC ← BC – 1 Repeat until A = (HL) or BC = 0		② ↕	X		X	① ↕			111 001	ED B9				If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES ① P/V flag … the result of BC … otherwise P/V …
② Z flag is 1 if A = (HL) otherwise Z = …

8-Bit Arithmetic and Logical Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
ADD A, r	A ← A + r			X		X	V			10 [000] r				4	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n			X		X	V			11 [000] 110 n					
ADD A, (HL)	A ← A + (HL)			X		X				10 [000] 110					
ADD A, (IX + d)	A ← A + (IX + d)			X		X				11 011 101 10 [000] 110 ← d →	DD				
ADD A, (IY + d)	A ← A + (IY + d)			X		X				11 111 101 10 [000] 110 ← d →	FD				
ADC A, s	A ← A + s + CY			X		X	V			[001]					s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction. The indicated bits replace the [000] in the ADD set above
SUB s	A ← A – s			X		X	V			[010]					
SBC A, s	A ← A – s – CY			X		X	V	1		[011]					
AND s	A ← A ∧ s			X		X	P	0		[100]					
OR s	A ← A ∨ s			X		X	P			[110]					
XOR s	A ← A ⊕ s			X		X	P			[101]					
CP s	A – s			X		X	V			[111]					
INC r	r ← r + 1			X		X	V	0		00 r [100]				4	
INC (HL)	(HL) ← (HL) + 1			X		X	V	0		00 110 [100]				11	
INC (IX + d)	(IX + d) ← (IX + d) + 1			X		X	V	0		11 011 101 00 110 [100] ← d →	DD			23	
INC (IY + d)	(IY + d) ← (IY + d) + 1			X		X	V	0		11 111 101 00 110 [100] ← d →	FD				
DEC m	m ← m – 1			X		X	V			[101]					m is any of r, (HL), (IX + d), (IY + d) as shown for INC. DEC same format and states as INC. Replace [100] with [101] in opcode

General-Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc content into packed BCD following add or subtract with packed BCD operands	I	I	X	I	X	P	•	I	00 100 111	27	1	1	4	Decimal adjust accumulator
CPL	A ← Ā			X						00 101 111	2F				Complement accumulator (one's complement)
NEG	A ← 0 A			X		X	V	I		11 101 101 01 000 100	ED 44	2	2	8	Negate acc (two's complement)
CCF	CY ← C̄Y			X	X	X				00 111 111	3F			4	Complement carry flag
SCF	CY ← 1			X	0	X			I	00 110 111	37	.		4	Set carry flag
NOP	No operation			X	•	X			•	00 000 000	00	1	:	4	
HALT	CPU halted			X	•	X			•	01 110 110	76	!	:	4	
DI ★	IFF ← 0			X	•	X			•	11 110 011	F3	1	1	4	
EI ★	IFF ← 1			X	•	X			•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0			X	•	X			•	11 101 101 01 000 110	ED 46	"	?	8	
IM 1	Set interrupt mode 1			X		X				11 101 101 01 010 110	ED 56			8	
IM 2	Set interrupt mode 2			X		X				11 101 101 01 011 110	ED 5E			8	

NOTES: IFF indicates the interrupt enable flip-flop
[illegible] indicates the carry flip-flop
★ indicates interrupts are not sampled at the end of EI or DI

16-Bit Arithmetic Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD HL ss	HL ← HL + ss	•	•	X	X	X	•	0		00 ss1 001		1	3	11	ss Reg 00 BC 01 DE 10 HL 11 SP
ADC HL ss	HL ← HL + ss + CY			X	X	X	V				ED		4		
SBC HL ss	HL ← HL ss CY			X	X	X	V							15	
ADD IX pp	IX ← IX +			X	X	X					DD			15	pp Reg 00 BC 01 DE 10 IX 11 SP
ADD IY rr	IY ← IY + rr			X	X	X								15	rr Reg 00 BC 01 DE 10 IY 11 SP
INC ss	ss +			X		X								6	
INC IX	IX ← IX + 1			X		X				[illegible]	DD 23			10	
INC IY	IY ← IY + 1			X		X				11 111 101 00 [illegible]	FD 23				
DEC ss	ss ← ss 1			X		X				00 ss1 011					
DEC IX	IX ← IX 1			X		X				101 011	DD 2B				
DEC IY	IY ← IY			X		X				11 00 101 011	FD 2B			10	

Rotate and Shift Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA				X		X				00					Rotate left circular accumulator
RLA	0 A			X		X									Rotate left accumulator
RRCA	7 → 0 → CY A			X		X									Rotate right circular accumulator
RRA	7 → 0 A					X									Rotate right accumulator
RL				X		X					F				Rotate left register
RLC (HL)						X	P				CB				Reg B [illegible] E H [illegible] A
RLC (IX + d)	CY ← 7 ← 0 HL (IX + (IY +					X	P				DD CB				
RLC (IY + d)						X					FD CB				
RL m	7 ← 0 m = r (HL (IX + d) (IY + d			X		X	P								Instruction format and states are shown for RLC. To form new opcode replace 000 or RLC shown
RRC m	7 → 0 → CY m = r (HL, (IX + d) (IY + d						P								

Rotate and Shift Group (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RR m	[7 → 0] → [CY] m ≡ r, (HL), (IX + d), (IY + d)			X	0	X	P	0		[011]					
SLA m	[CY] ← [7 ← 0] ← 0 m ≡ r, (HL), (IX + d), (IY + d)			X	0	X	P			[100]					
SRA m	[7 → 0] → [CY] m ≡ r, (HL), (IX + d), (IY + d)			X	0	X	P			[101]					
SRL m	0 → [7 → 0] → [CY] m ≡ r, (HL), (IX + d), (IY + d)			X		X	P			[111]					
RLD	[7 - 4 | 3 - 0] A [7 - 4 | 3 - 0] (HL)			X		X	P			11 101 101 01 101 111	ED 6F				Rotate digit left and right between the accumulator and location (HL).
RRD	[7 - 4 | 3 - 0] (HL)			X		X	P			11 101 101 01 100 111	ED 67				The content of the upper half of the accumulator is unaffected

Bit Set, Reset and Test Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	Z ← $\overline{r}_b$	X		X	1	X	X	0		11 001 011 01 b r	CB	2	2	8	r Reg. 000 B
BIT b, (HL)	Z ← $\overline{(HL)}_b$	X		X		X	X	0		11 001 011 01 b 110	CB	2	3	12	001 C 010 D
BIT b, (IX + d)b	Z ← $\overline{(IX+d)}_b$	X		X	1	X	X	0		11 011 101 11 001 011 – d – 01 b 110	DD CB	4	5	20	011 E 100 H 101 L 111 A b Bit Tested
BIT b, (IY + d)b	Z ← $\overline{(IY+d)}_b$	X		X		X	X	0		11 111 101 11 001 011 – d – 01 b 110	FD CB			20	000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
SET b, r	r_b ← 1			X		X				11 001 011 [11] b r	CB		2	8	
SET b, (HL)	$(HL)_b$ ← 1			X		X				11 001 011 [11] b 110	CB	2			
SET b, (IX + d)	$(IX+d)_b$ ← 1			X		X				11 011 101 11 001 011 – d – [11] b 110	DD CB			23	
SET b, (IY + d)	$(IY+d)_b$ ← 1			X		X				11 111 101 11 001 011 – d – [11] b 110	FD CB		6	23	
RES b, m	m_b ← 0 m ≡ r, (HL), (IX + d), (IY + d)			X		X				[10]					To form new opcode replace [11] of SET b, s with [10]. Flags and time states for SET instruction.

NOTES: The notation m_b indicates bit b (0 to 7) or location m.

Jump Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•		X	•	X				11 000 011 – n – – n –	C3	3	3	10	cc Condition
JP cc	If condition cc is true PC ← nn, otherwise continue			X	•	X				11 cc 010 – n – – n –		3	3	10	000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive
JR e	PC ← PC + e			X		X				00 011 000 – e - 2 –	18	2	3	12	111 M sign negative
JR C, e	If C = 0, continue			X	•	X				00 111 000 – e - 2 –	38	2	2	7	If condition not met
	If C = 1, PC ← PC + e											2	3	12	If condition is met.
JR NC, e	If C = 1, continue			X	•	X				00 110 000 – e - 2 –	30	2	2	7	If condition not met
	If C = 0, PC ← PC + e											2	3	12	If condition is met
JP Z, e	If Z = 0 continue			X	•	X				00 101 000 – e - 2 –	28	2		7	If condition not met
	If Z = 1, PC ← PC + e											2	3	12	If condition is met
JR NZ, e	If Z = 1 continue			X		X	•			00 100 000 – e - 2 –	20	2	2	7	If condition not met
	If Z = 0, PC ← PC + e											2	3	12	If condition is met
JP (HL)	PC ← HL			X	•	X				11 101 001	E9			4	
JP (IX)	PC ← IX			X	•	X				11 011 101 11 101 001	DD E9	2	2	8	

Jump Group (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B−1 If B = 0, continue		•	X	•	X	•			00 010 000 ← e−2 →	10	2	2	8	If B = 0.
	If B ≠ 0, PC ← PC+e											2	3	13	If B ≠ 0.

NOTES e represents the extension in the relative addressing mode
e is a signed two's complement number in the range < −126, 129 >
e−2 in the opcode provides an effective address of pc+e as PC is incremented by 2 prior to the addition of e

Call and Return Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	(SP−1) ← PC_H (SP−2) ← PC_L PC ← nn	•	•	X	•	X	•	•	•	11 001 101 ← n → ← n →	CD	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn		•	X	•	X				11 cc 100 ← n → ← n →		3 3	3 5	10 17	If cc is false. If cc is true.
RET	PC_L ← (SP) PC_H ← (SP+1)	•	•	X	•	X				11 001 001	C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET		•	X	•	X				11 cc 000		 1	1 3	5 11	If cc is false If cc is true. cc Condition 000 NZ non-zero 001 Z zero 010 NC non carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
RETI	Return from interrupt		•	X	•	X	•			11 101 101 01 001 101	ED 4D			14	
RETN[1]	Return from non maskable interrupt			X		X				11 101 101 01 000 101	ED 45	2		14	
RST p	(SP−1) ← PC_H (SP−2) ← PC_L PC_H ← 0 PC_L ← p		•	X		X				11 t 111			3	11	t p 000 00H 001 08H 010 10H 011 18H 100 20H 101 28H 110 30H 111 38H

NOTE [1]RETN loads IFF_2 ← IFF_1

Input and Output Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)			X	•	X				11 011 011 ← n →	DB	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) if r = 110 only the flags will be affected			X		X	P	0		11 101 101 01 r 000	ED		3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B−1 HL ← HL + 1	X	① 1	X	X	X	X			11 101 101 10 100 010	ED A2				C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B−1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1		11 101 101 10 110 010	ED B2	2	5 (If B≠0) 4 (If B=0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B − 1 HL ← HL − 1	X	① 1	X	X	X	X			11 101 101 10 101 010	ED AA			16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B−1 HL ← HL−1 Repeat until B = 0	X	1	X	X	X	X			11 101 101 10 111 010	ED BA	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A			X		X				11 010 011 ← n →	D3	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r		•	X		X				11 101 101 01 r 001	ED		3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B−1 HL ← HL + 1	X	① 1	X	X	X	X	1		11 101 101 10 100 011	ED A3				C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B−1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X			11 101 101 10 110 011	ED B3	 2	5 (If B≠0) 4 (If B=0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B−1 HL ← HL − 1	X	① 1	X	X	X	X	1		11 101 101 10 101 011	ED AB	2		16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

NOTE ① If the result of B − 1 is zero the Z flag is set, otherwise it is reset

Input and Output Group (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	↕	X	X	X	X	1	•	101 101 ED 111 011		5 (If B≠0) 4 (If B = 0)	21	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

Summary of Flag Operation

Instruction	D_7 S	Z		H		P/V	N	D_0 C	Comments
ADD A, s, ADC A, s	↕	↕	X	↕	X	V	0	↕	8 bit add or add with carry
SUB s, SBC A, s, CP s, NEG	↕	↕	X	↕	X	V	1	↕	8 bit subtract, subtract with carry
AND s	↕	↕	X	1	X	P	0	0	Logical operations
OR s, XOR s	↕	↕	X	0	X	P	0	0	Logical operations
INC s	↕	↕	X	↕	X	V		•	8 bit increment
DEC s	↕	↕	X	↕	X	V		•	8 bit decrement
ADD DD, ss	•	•	X	X	X	•		↕	16 bit add
ADC HL, ss	↕	↕	X	X	X	V		↕	16 bit add with carry
SBC HL, ss	↕	↕	X	X	X	V		↕	16 bit subtract with carry
RLA, RLCA, RRA, RRCA	•	•	X	0	X	•		↕	Rotate accumulator
RL m, RLC m, RR m, RRC m, SLA m, SRA m, SRL m	↕		X	0	X	P		↕	Rotate and shift [illegible]
RLD, RRD	↕	↕	X		X	P			Rotate digit [illegible]
DAA	↕	↕	X		X	P		↕	Decimal adjust [illegible]
CPL	•	•	X	1	X	•		•	Complement [illegible]
SCF	•	•	X	0	X	•		1	Set carry
CCF	•	•	X	X	X	•		↕	Complement [illegible]
IN r (C)	↕	↕	X	0	X	P		•	Input register [illegible]
INI, IND, OUTI, OUTD	X	↕	X	X	X	X		•	Block [illegible] Z = 0 if B ≠ [illegible]
INIR, INDR, OTIR, OTDR	X	↕	X	X	X	X		•	Block [illegible] Z = 0 if B ≠ [illegible]
LDI, LDD	X	X	X	0	X	↕		•	Block [illegible] P/V [illegible] if BC ≠ 0, otherwise P/V = 0
LDIR, LDDR	X	X	X		X	0		•	Block [illegible] P/V [illegible] if BC ≠ 0, otherwise P/V = 0
CPI, CPIR, CPD, CPDR	X	↕	X			↕		•	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V [illegible] if BC ≠ [illegible] otherwise P/V = 0.
LD A, I, LD A, R	↕		X		X	IFF			The [illegible] of the interrupt enable flip-flop (IFF) is copied [illegible] the P/V flag.
BIT b, s	X		X		X	X			The state of bit b of location s is copied into the Z flag

Symbolic Notation

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1.
Z	Zero flag. Z = 1 if the result of the operation is 0.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.

Symbol	Operation
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
	Any one of the CPU registers A, B, C, D, E, H, L.
	Any 8-bit location for all the addressing modes allowed for the particular instruction.
	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8 bit value in range < 0, 255 >.
nn	16-bit value in range < 0, 65535 >.

Pin Descriptions

A_0-A_{15}. *Address Bus* (output, active High, 3-state). A_0-A_{15} form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.

$\overline{BUSACK}$. *Bus Acknowledge* (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ have entered their high impedance states. The external circuitry can now control these lines.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). Bus Request has a higher priority than $\overline{NMI}$ and is always recognized at the end of the current machine cycle. $\overline{BUSREQ}$ forces the CPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ to go to a high-impedance state so that other devices can control these lines. $\overline{BUSREQ}$ is normally wire-ORed and requires an external pullup for these applications. Extended $\overline{BUSREQ}$ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.

D_0-D_7. *Data Bus* (input/output, active High, 3-state). D_0-D_7 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

$\overline{HALT}$. *Halt State* (output, active Low). $\overline{HALT}$ indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.

$\overline{INT}$. *Interrupt Request* (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. $\overline{INT}$ is normally wire-ORed and requires an external pullup for these applications.

$\overline{IORQ}$. *Input/Output Request* (output, active Low, 3-state). $\overline{IORQ}$ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. $\overline{IORQ}$ is also generated concurrently with $\overline{M1}$ during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

$\overline{M1}$. *Machine Cycle One* (output, active Low). $\overline{M1}$, together with $\overline{MREQ}$, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. $\overline{M1}$, together with $\overline{IORQ}$, indicates an interrupt acknowledge cycle.

$\overline{MREQ}$. *Memory Request* (output, active Low, 3-state). $\overline{MREQ}$ indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{NMI}$. *Non-Maskable Interrupt* (input, active Low). $\overline{NMI}$ has a higher priority than $\overline{INT}$. $\overline{NMI}$ is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

$\overline{RD}$. *Memory Read* (output, active Low, 3-state). $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that $\overline{RESET}$ must be active for a minimum of three full clock cycles before the reset operation is complete.

$\overline{RFSH}$. *Refresh* (output, active Low). $\overline{RFSH}$, together with $\overline{MREQ}$, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

$\overline{WAIT}$. *Wait* (input, active Low). $\overline{WAIT}$ indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended $\overline{WAIT}$ periods can prevent the CPU from refreshing dynamic memory properly.

$\overline{WR}$. *Memory Write* (output, active Low, 3-state). $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.



8-Bit Microprocessor Family

SY6500

MICROPROCESSOR PRODUCTS

- Single 5 V ±5% power supply
- N channel, silicon gate, depletion load technology
- Eight bit parallel processing
- 56 Instructions
- Decimal and binary arithmetic
- Thirteen addressing modes
- True indexing capability
- Programmable stack pointer
- Variable length stack
- Interrupt capability
- Non-maskable interrupt
- Use with any type or speed memory
- Bi-directional Data Bus
- Instruction decoding and control
- Addressable memory ranqe of up to 65 K bytes
- "Ready" input
- Direct memory access capability
- Bus compatible with MC6800
- Choice of external or on-board clocks
- 1 MHz, 2 MHz, 3 MHz and 4 MHz operation
- On-chip clock options
  - * External single clock input
  - * Crystal time base input
- 40 and 28 pin package versions
- Pipeline architecture

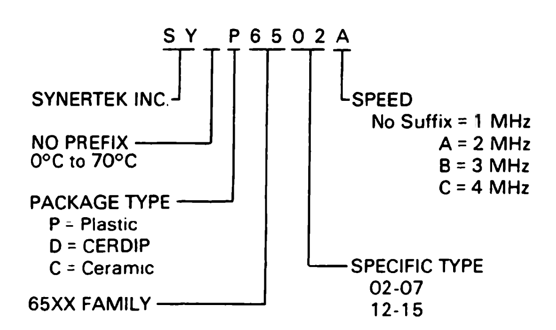
The SY6500 Series Microprocessors represent the first totally software compatible microprocessor family. This family of products includes a range of software compatible microprocessors which provide a selection of addressable memory range, interrupt input options and on-chip clock oscillators and drivers. All of the microprocessors in the SY6500 family are software compatible within the group and are bus compatible with the MC6800 product offering.

The family includes six microprocessors with on-board clock oscillators and drivers and four microprocessors driven by external clocks. The on-chip clock versions are aimed at high performance, low cost applications where single phase inputs or crystals provide the time base. The external clock versions are geared for the multi-processor system applications where maximum timing control is mandatory. All versions of the microprocessors are available in 1 MHz, 2 MHz, 3 MHz and 4 MHz maximum operating frequencies.

MEMBERS OF THE FAMILY

PART NUMBERS	CLOCKS	PINS	$\overline{IRQ}$	$\overline{NMI}$	RYD	ADDRESSING
SY6502	On-Chip	40	√	√	√	64 K
SY6503	"	28	√	√		4 K
SY6504		28	√			8 K
SY6505		28	√		√	4 K
SY6506		28	√			4 K
SY6507		28			√	8 K
SY6512	External	40	√	√	√	64 K
SY6513	"	28	√	√		4 K
SY6514		28	√			8 K
SY6515		28	√		√	4 K

ORDERING INFORMATION



Only 6502 and 6512 are available in 3 and 4 MHz

COMMENTS ON THE DATA SHEET

The data sheet is constructed to review first the basic "Common Characteristics" – those features which are common to the general family of microprocessors. Subsequent to a review of the family characteristics will be sections devoted to each member of the group with specific features of each.

SY6500 INTERNAL ARCHITECTURE

REGISTER SECTION | CONTROL SECTION

$\overline{RES}$ $\overline{IRQ}$ $\overline{NMI}$

INTERRUPT LOGIC

INSTRUCTION DECODE

RDY

TIMING CONTROL

ADDRESS BUS

AB0 AB1 AB2 AB3 AB4 AB5 AB6 AB7

ABL

AB8 AB9 AB10 AB11 AB12 AB13 AB14 AB15

ABH

INTERNAL ADH

INTERNAL ADL

INTERNAL DATA BUS

INDEX REGISTER Y

INDEX REGISTER X

STACK POINTER REGISTER (S)

ACCUMULATOR A

PCL

PCH

INPUT DATA LATCH (DL)

DATA BUS BUFFER

PROCESSOR STATUS REGISTER P

INSTRUCTION REGISTER

CLOCK GENERATOR

$\emptyset_1$ $\emptyset_2$

$\emptyset_1$ (IN)

$\emptyset_2$ (IN)

SY651X

CLOCK INPUT

SY650X

$\emptyset_1$ OUT

$\emptyset_2$ OUT

R/$\overline{W}$

DBE

DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7

DATA BUS

LEGEND:

= 8 BIT LINE

NOTE

1. CLOCK GENERATOR IS NOT INCLUDED ON SY651X
2. ADDRESSING CAPABILITY AND CONTROL OPTIONS VARY WITH EACH OF THE SY6500 PRODUCTS.

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	V
Input Voltage	V_{in}	-0.3 to +7.0	V
Operating Temperature	T_A	0 to +70	°C
Storage Temperature	T_{STG}	-55 to +150	°C

COMMENT

This device contains input protection against damage due to high static voltages or electric fields; however, precautions should be taken to avoid application of voltages higher than the maximum rating.

D.C. CHARACTERISTICS (V_{CC} = 5.0V ±5%, T_A = 0-70°C)
($Ø_1$, $Ø_2$ applies to SY651X, $Ø_{o(in)}$ applies to SY650X)

Symbol	Characteristic	Min.	Max.	Unit
V_{IH}	Input High Voltage Logic and $Ø_o$ (in) for all 650X devices — 1,2,3 MHz Logic and $Ø_o$ (in) for all 650X devices — 4 MHz $Ø_1$ and $Ø_2$ only for all 651X devices. Logic as 650X — All Speeds	 +2.4 +3.3 V_{CC} -0.5	 V_{CC} V_{CC} V_{CC} + 0.25	 V V V
V_{IL}	Input Low Voltage Logic, $Ø_{o(in)}$ (650X) $Ø_1$, $Ø_2$ (651X)	 -0.3 -0.3	 +0.4 +0.2	 V
I_{IL}	Input Loading (V_{in} = 0 V, V_{CC} = 5.25 V) RDY, S.O.	-10	-300	µA
I_{in}	Input Leakage Current (V_{in} = 0 to 5.25 V, V_{CC} = 0) Logic (Excl. RDY, S.O.) $Ø_1$, $Ø_2$ (651X) $Ø_{o(in)}$ (650X)		 2.5 100 10.0	 µA µA µA
I_{TSI}	Three-State (Off State) Input Current (V_{in} = 0.4 to 2.4 V, V_{CC} = 5.25 V) DB0-DB7		±10	µA
V_{OH}	Output High Voltage (I_{LOAD} = -100µAdc, V_{CC} = 4.75 V) 1,2,3 MHz SYNC, DB0-DB7, A0-A15, R/$\overline{W}$ 4 MHz	 2.4 2.0	 – –	 V V
V_{OL}	Output Low Voltage (I_{LOAD} = 1.6mAdc, V_{CC} = 4.75 V) 1,2,3 MHz SYNC, DB0-DB7, A0-A15, R/$\overline{W}$ 4 MHz	 – –	 0.4 0.8	 V V
P_D	Power Dissipation 1 MHz and 2 MHz (V_{CC} = 5.25V) 3 MHz 4 MHz	– – –	700 800 900	mW mW mW
C	Capacitance (V_I = 0, T_A = 25°C, f = 1 MHz)			
C_{in}	$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, S.O., DBE	–	10	pF
	DB0-DB7	–	15	pF
C_{out}	A0-A15, R/$\overline{W}$, SYNC		12	pF
$C_{Ø_{o(in)}}$	$Ø_{o(in)}$ (650X)	–	15	pF
$C_{Ø_1}$	$Ø_1$ (651X)	–	50	pF
$C_{Ø_2}$	$Ø_2$ (651X)	–	80	pF

TIMING DEFINITIONS, COMPOSITE TIMING DIAGRAM (See note at bottom of page 5)

SY651X INPUT CLOCK TIMING

SY650X INPUT CLOCK TIMING

SY65XX TIMING (See note at bottom of page 5)

DYNAMIC OPERATING CHARACTERISTICS

(V_{CC} = 5.0 ± 5%, T_A = 0° to 70°C)

Parameter	Symbol	1 MHz		2 MHz		3 MHz		4 MHz		Units
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
651X										
Cycle Time	T_{CYC}	1.00	40	0.50	40	0.33	40	0.25	40	µs
$\emptyset_1$ Pulse Width	$T_{PWH\emptyset_1}$	430	—	215	—	150	—			ns
$\emptyset_2$ Pulse Width	$T_{PWH\emptyset_2}$	470	—	235	—	160	—			ns
Delay Between $\emptyset_1$ and $\emptyset_2$	T_D	0	—	0	—	0	—			ns
$\emptyset_1$ and $\emptyset_2$ Rise and Fall Times[1]	T_R, T_F	0	25	0	20	0	15			ns
650X										
Cycle Time	T_{CYC}	1.00	40	0.50	40	0.33	40	0.25	40	µs
$\emptyset_{o(IN)}$ Low Time[2]	$T_{L\emptyset_o}$	480	—	240	—	160	—	110	—	ns
$\emptyset_{o(IN)}$ High Time[2]	$T_{H\emptyset_o}$	460	—	240	—	160	—	115	—	ns
$\emptyset_o$ Neg to $\emptyset_1$ Pos Delay[5]	T_{01+}	10	70	10	70	10	70	10	70	ns
$\emptyset_o$ Neg to $\emptyset_2$ Neg Delay[5]	T_{02-}	5	65	5	65	5	65	5	65	ns
$\emptyset_o$ Pos to $\emptyset_1$ Neg Delay[5]	T_{01}	5	65	5	65	5	65	5	65	ns
$\emptyset_o$ Pos to $\emptyset_2$ Pos Delay[5]	T_{02+}	15	75	15	75	15	75	15	75	ns
$\emptyset_{o(IN)}$ Rise and Fall Time[1]	T_{RO}, T_{FO}	0	30	0	20	0	15	0	10	ns
$\emptyset_{1(OUT)}$ Pulse Width	$T_{PWH\emptyset_1}$	$T_{L\emptyset_o}$-20	$T_{L\emptyset_o}$	$T_{L\emptyset_o}$-20	$T_{L\emptyset_o}$	$T_{L\emptyset_o}$-20	$T_{L\emptyset_o}$	$T_{L\emptyset_o}$-20	$T_{L\emptyset_o}$	ns
$\emptyset_{2(OUT)}$ Pulse Width	$T_{PWH\emptyset_2}$	$T_{L\emptyset_o}$-40	$T_{L\emptyset_o}$-10	$T_{L\emptyset_o}$-40	$T_{L\emptyset_o}$-10	$T_{L\emptyset_o}$-40	$T_{L\emptyset_o}$-10	$T_{L\emptyset_o}$-40	$T_{L\emptyset_o}$-10	ns
Delay Between $\emptyset_1$ and $\emptyset_2$	T_D	5	—	5	—	5	—	5	—	ns
$\emptyset_1$ and $\emptyset_2$ Rise and Fall Times[1,3]	T_R, T_F	—	25	—	25	—	15	—	15	ns
650X, 651X										
R/$\overline{W}$ Setup Time	T_{RWS}	—	225	—	140	—	110	—	90	ns
R/$\overline{W}$ Hold-Time	T_{RWH}	30	—	30	—	15	—	10	—	ns
Address Setup Time	T_{ADS}	—	225	—	140	—	110	—	90	ns
Address Hold Time	T_{ADH}	30	—	30	—	15	—	10	—	ns
Read Access Time	T_{ACC}	—	650	—	310	—	170	—	110	ns
Read Data Setup Time	T_{DSU}	100	—	50	—	50	—	50	—	ns
Read Data Hold Time	T_{HR}	10	—	10	—	10	—	10	—	ns
Write Data Setup Time	T_{MDS}	20	175	20	100	20	75	—	70	ns
Write Data Hold Time	T_{HW}	60	150	60	150	30	130	20	—	ns
Sync Setup Time	T_{SYS}	—	350	—	175	—	100	—	90	ns
Sync Hold Time	T_{SYH}	30	—	30	—	15	—	15	—	ns
RDY Setup Time[4]	T_{RS}	200	—	200	—	150	—	120	—	ns

NOTES:

1. Measured between 10% and 90% points on waveform.
2. Measured at 50% points.
3. Load = 1 TTL load +30 pF.
4. RDY must never switch states within T_{RS} to end of $\emptyset_2$.
5. Load = 100 pF.
6. The 2 MHz devices are identified by an "A" suffix.
7. The 3 MHz devices are identified by a "B" suffix.
8. The 4 MHz devices are identified by a "C" suffix.

TIMING DIAGRAM NOTE:

Because the clock generation for the SY650X and SY651X is different, the two clock timing sections are referenced to the main timing diagram by three reference lines marked REF 'A', REF 'B' and REF 'C'. Reference between the two sets of clock timings is without meaning. Timing parameters are referred to these lines and scale variations in the diagrams are of no consequence.

PIN FUNCTIONS

Clocks ($Ø_1$, $Ø_2$)

The SY651X requires a two phase non-overlapping clock that runs at the V_{CC} voltage level.

The SY650X clocks are supplied with an internal clock generator. The frequency of these clocks is externally controlled. Clock generator circuits are shown elsewhere in this data sheet.

Address Bus (A_0-A_{15}) (See sections on each micro for respective address lines on those devices.)

These outputs are TTL compatible, capable of driving one standard TTL load and 130 pF.

Data Bus (DB_0-DB_7)

Eight pins are used for the data bus. This is a bi-directional bus, transferring data to and from the device and peripherals. The outputs are three-state buffers, capable of driving one standard TTL load and 130 pF.

Data Bus Enable (DBE)

This TTL compatible input allows external control of the three-state data output buffers and will enable the microprocessor bus driver when in the high state. In normal operation DBE would be driven by the phase two ($Ø_2$) clock, thus allowing data output from microprocessor only during $Ø_2$. During the read cycle, the data bus drivers are internally disabled, becoming essentially an open circuit. To disable data bus drivers externally, DBE should be held low. This signal is available on the SY6512, only.

Ready (RDY)

This input signal allows the user to halt the microprocessor on all cycles except write cycles. A negative transition to the low state during or coincident with phase one ($Ø_1$) will halt the microprocessor with the output address lines reflecting the current address being fetched. This condition will remain through a subsequent phase two ($Ø_2$) in which the Ready signal is low. This feature allows microprocessor interfacing with low speed PROMS as well as fast (max. 2 cycle) Direct Memory Access (DMA). If ready is low during a write cycle, it is ignored until the following read operation. Ready transitions must not be permitted during $Ø_2$ time.

Interrupt Request ($\overline{IRQ}$)

This TTL level input requests that an interrupt sequence begin within the microprocessor. The microprocessor will complete the current instruction being executed before recognizing the request. At that time, the interrupt mask bit in the Status Code Register will be examined. If the interrupt mask flag is not set, the microprocessor will begin an interrupt sequence. The Program Counter and Processor Status Register are stored in the stack. The microprocessor will then set the interrupt mask flag high so that no further interrupts may occur. At the end of this cycle, the program counter low will be loaded from address FFFE, and program counter high from location FFFF, therefore transferring program control to the memory vector located at these addresses. The RDY signal must be in the high state for any interrupt to be recognized. A 3KΩ external resistor should be used for proper wire-OR operation.

Non-Maskable Interrupt ($\overline{NMI}$)

A negative going transition on this input requests that a non-maskable interrupt sequence be generated within the microprocessor.

$\overline{NMI}$ is an unconditional interrupt. Following completion of the current instruction, the sequence of operations defined for $\overline{IRQ}$ will be performed, regardless of the state interrupt mask flag. The vestor address loaded into the program counter, low and high, are locations FFFA and FFFB respectively, thereby transferring program control to the memory vector located at these addresses. The instructions loaded at these locations cause the microprocessor to branch to a non-maskable interrupt routine in memory.

$\overline{NMI}$ also requires an external 3KΩ resistor to V_{CC} for proper wire-OR operations.

Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupts lines that are sampled during $Ø_2$ (phase 2) and will begin the appropriate interrupt routine on the $Ø_1$ (phase 1) following the completion of the current instruction.

Set Overflow Flag (S.O.)

A NEGATIVE going edge on this input sets the overflow bit in the Status Code Register. This signal is sampled on the trailing edge of $Ø_1$.

SYNC

This output line is provided to identify those cycles in which the microprocessor is doing an OP CODE fetch. The SYNC line goes high during $Ø_1$ of an OP CODE fetch and stays high for the remainder of that cycle. If the RDY line is pulled low during the $Ø_1$ clock pulse in which SYNC went high, the processor will stop in its current state and will remain in the state until the RDY line goes high. In this manner, the SYNC signal can be used to control RDY to cause single instruction execution.

Reset ($\overline{RES}$)

This input is used to reset or start the microprocessor from a power down condition. During the time that this line is held low, writing to or from the microprocessor is inhibited. When a positive edge is detected on the input, the microprocessor will immediately begin the reset sequence.

After a system initialization time of six clock cycles, the mask interrupt flag will be set and the microprocessor will load the program counter from the memory vector locations FFFC and FFFD. This is the start location for program control.

After V_{CC} reaches 4.75 volts in a power up routine, reset must be held low for at least two clock cycles. At this time the R/$\overline{W}$ and SYNC signal will become valid.

When the reset signal goes high following these two clock cycles, the microprocessor will proceed with the normal reset procedure detailed above.

Read/Write (R/$\overline{W}$)

This output signal is used to control the direction of data transfers between the processor and other circuits on the data bus. A high level on R/$\overline{W}$ signifies data into the processor; a low is for data transfer out of the processor.

PROGRAMMING CHARACTERISTICS

INSTRUCTION SET – ALPHABETIC SEQUENCE

Mnemonic	Description
ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EOR	Memory with Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift One Bit Right (Memory or
NOP	No Operation
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

ADDRESSING MODES

Accumulator Addressing

This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

Immediate Addressing

In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

Absolute Addressing

In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

Zero Page Addressing

The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

Indexed Zero Page Addressing – (X, Y indexing)

This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y." The effective address is calcuated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

Indexed Absolute Addressing – (X, Y indexing)

This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X," and "Absolute, Y." The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

Implied Addressing

In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

Relative Addressing

Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is –128 to +127 bytes from the next instruction.

Indexed Indirect Addressing

In indexed indirect addressing (referred to as (Indirect,X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents is the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

Indirect Indexed Addressing

In indirect indexed addressing (referred to as (Indirect),Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

Absolute Indirect

The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address. The next memory location contains the high order byte of the effective address which is loaded into the sixteen bits of the program counter.

PROGRAMMING CHARACTERISTICS

PROGRAMMING MODEL

Bits	Register	
7 … ø	A	ACCUMULATOR
7 … ø	Y	INDEX REGISTER
7 … ø	X	INDEX REGISTER
15 … 7 … ø	PCH / PCL	PROGRAM COUNTER
7 … ø	S	STACK POINTER

PROCESSOR STATUS REG "P" (bits 7 … ø): N V – B D I Z C

Bit	Flag	
C	CARRY	1 = TRUE
Z	ZERO	1 = RESULT ZERO
I	IRQ DISABLE	1 = DISABLE
D	DECIMAL MODE	1 = TRUE
B	BRK COMMAND	1 = BRK
V	OVERFLOW	1 = TRUE
N	NEGATIVE	1 = NEG.

INSTRUCTION SET — OP CODES, EXECUTION TIME, MEMORY REQUIREMENTS

(Each addressing-mode cell gives OP N #.)

MNEMONIC	OPERATION		IMMEDIATE	ABSOLUTE	ZERO PAGE	ACCUM	IMPLIED	(IND, X)	(IND), Y	Z PAGE, X	ABS, X	ABS, Y	RELATIVE	INDIRECT	Z PAGE, Y	N	Z	C	I	D	V
ADC	A + M + C → A	(1)	69 2 2	6D 4 3	65 3 2			61 6 2	71 5 2	75 4 2	7D 4 3	79 4 3				✓	✓	✓			✓
AND	A ∧ M → A	(1)	29 2 2	2D 4 3	25 3 2			21 6 2	31 5 2	35 4 2	3D 4 3	39 4 3				✓	✓				
ASL	C ← [7 … 0] ← 0			0E 6 3	06 5 2	0A 2 1				16 6 2	1E 7 3					✓	✓	✓			
BCC	BRANCH ON C = 0	(2)											90 2 2								
BCS	BRANCH ON C = 1	(2)											B0 2 2								
BEQ	BRANCH ON Z = 1	(2)											F0 2 2								
BIT	A ∧ M			2C 4 3	24 3 2											M_7	✓				M_6
BMI	BRANCH ON N = 1	(2)											30 2 2								
BNE	BRANCH ON Z = 0	(2)											D0 2 2								
BPL	BRANCH ON N = 0	(2)											10 2 2								
BRK	(See Fig. 1)						00 7 1														
BVC	BRANCH ON V = 0	(2)											50 2 2								
BVS	BRANCH ON V = 1	(2)											70 2 2								
CLC	0 → C						18 2 1											0			
CLD	0 → D						D8 2 1													0	
CLI	0 → I						58 2 1												0		
CLV	0 → V						B8 2 1														0
CMP	A − M	(1)	C9 2 2	CD 4 3	C5 3 2			C1 6 2	D1 5 2	D5 4 2	DD 4 3	D9 4 3				✓	✓	✓			
CPX	X − M		E0 2 2	EC 4 3	E4 3 2											✓	✓	✓			
CPY	Y − M		C0 2 2	CC 4 3	C4 3 2											✓	✓	✓			
DEC	M − 1 → M			CE 6 3	C6 5 2					D6 6 2	DE 7 3					✓	✓				
DEX	X − 1 → X						CA 2 1									✓	✓				
DEY	Y − 1 → Y						88 2 1									✓	✓				
EOR	A ⊻ M → A	(1)	49 2 2	4D 4 3	45 3 2			41 6 2	51 5 2	55 4 2	5D 4 3	59 4 3				✓	✓				
INC	M + 1 → M			EE 6 3	E6 5 2					F6 6 2	FE 7 3					✓	✓				
INX	X + 1 → X						E8 2 1									✓	✓				
INY	Y + 1 → Y						C8 2 1									✓	✓				
JMP	JUMP TO NEW LOC			4C 3 3										6C 5 3							
JSR	(See Fig. 2) JUMP SUB			20 6 3																	
LDA	M → A	(1)	A9 2 2	AD 4 3	A5 3 2			A1 6 2	B1 5 2	B5 4 2	BD 4 3	B9 4 3				✓	✓				
LDX	M → X	(1)	A2 2 2	AE 4 3	A6 3 2							BE 4 3			B6 4 2	✓	✓				
LDY	M → Y	(1)	A0 2 2	AC 4 3	A4 3 2					B4 4 2	BC 4 3					✓	✓				
LSR	0 → [7 … 0] → C			4E 6 3	46 5 2	4A 2 1				56 6 2	5E 7 3					0	✓	✓			
NOP	NO OPERATION						EA 2 1														
ORA	A ∨ M → A		09 2 2	0D 4 3	05 3 2			01 6 2	11 5 2	15 4 2	1D 4 3	19 4 3				✓	✓				
PHA	A → Ms S − 1 → S						48 3 1														
PHP	P → Ms S − 1 → S						08 3 1														
PLA	S + 1 → S Ms → A						68 4 1									✓	✓				
PLP	S + 1 → S Ms → P						28 4 1									(RESTORED)					
ROL	← [7 … 0] ← [C] ←			2E 6 3	26 5 2	2A 2 1				36 6 2	3E 7 3					✓	✓	✓			
ROR	→ [C] → [7 … 0] →			6E 6 3	66 5 2	6A 2 1				76 6 2	7E 7 3					✓	✓	✓			
RTI	(See Fig. 1) RTRN INT						40 6 1									(RESTORED)					
RTS	(See Fig. 2) RTRN SUB						60 6 1														
SBC	A − M − C̄ → A	(1)	E9 2 2	ED 4 3	E5 3 2			E1 6 2	F1 5 2	F5 4 2	FD 4 3	F9 4 3				✓	✓	✓			✓
SEC	1 → C						38 2 1											1			
SED	1 → D						F8 2 1													1	
SEI	1 → I						78 2 1												1		
STA	A → M			8D 4 3	85 3 2			81 6 2	91 6 2	95 4 2	9D 5 3	99 5 3				–	–	–	–	–	–
STX	X → M			8E 4 3	86 3 2										96 4 2	–	–	–	–	–	–
STY	Y → M			8C 4 3	84 3 2					94 4 2						–	–	–	–	–	–
TAX	A → X						AA 2 1									✓	✓	–	–	–	–
TAY	A → Y						A8 2 1									✓	✓	–	–	–	–
TSX	S → X						BA 2 1									✓	✓	–	–	–	–
TXA	X → A						8A 2 1									✓	✓	–	–	–	–
TXS	X → S						9A 2 1									–	–	–	–	–	–
TYA	Y → A						98 2 1									✓	✓	–	–	–	–

(1) ADD 1 TO N IF PAGE BOUNDARY IS CROSSED
(2) ADD 1 TO N IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO N IF BRANCH OCCURS TO DIFFERENT PAGE
(3) CARRY NOT = BELOW
(4) IF IN DECIMAL MODE Z FLAG IS INVALID
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

X INDEX X	+ ADD	– NOT MODIFIED
Y INDEX Y	− SUBTRACT	M_7 MEMORY BIT 7
A ACCUMULATOR	∧ AND	M_6 MEMORY BIT 6
M MEMORY PER EFFECTIVE ADDRESS	∨ OR	N NO. CYCLES
Ms MEMORY PER STACK POINTER	⊻ EXCLUSIVE OR	# NO. BYTES
	✓ MODIFIED	

SY6502 — 40 Pin Package

Signal	Pin	Pin	Signal
V_{SS}	1	40	$\overline{RES}$
RDY	2	39	$Ø_2$ (OUT)
$Ø_1$ (OUT)	3	38	S.O.
$\overline{IRQ}$	4	37	$Ø_0$ (IN)
N.C.	5	36	N.C.
$\overline{NMI}$	6	35	N.C.
SYNC	7	34	R/$\overline{W}$
V_{CC}	8	33	DB0
AB0	9	32	DB1
AB1	10	31	DB2
AB2	11	30	DB3
AB3	12	29	DB4
AB4	13	28	DB5
AB5	14	27	DB6
AB6	15	26	DB7
AB7	16	25	AB15
AB8	17	24	AB14
AB9	18	23	AB13
AB10	19	22	AB12
AB11	20	21	V_{SS}

Features

- 65K Addressable Bytes of Memory
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- On-the-chip Clock
  - √ TTL Level Single Phase Input
  - √ Crystal Time Base Input
- SYNC Signal
  (can be used for single instruction execution)
- RDY Signal
  (can be used for single cycle execution)
- Two Phase Output Clock for Timing of Support Chips



Asynchronous Communication Interface Adapter

SYNERTEK

A SUBSIDIARY OF HONEYWELL

SY6551

MICROPROCESSOR PRODUCTS

- On-chip baud rate generator: 15 programmable baud rates derived from a standard 1.8432 MHz external crystal (50 to 19,200 baud).
- Programmable interrupt and status register to simplify software design.
- Single +5 volt power supply.
- Serial echo mode.
- False start bit detection.
- 8-bit bi-directional data bus for direct communication with the microprocessor.
- External 16x clock input for non-standard baud rates (up to 125 Kbaud).
- Programmable: word lengths; number of stop bits; and parity bit generation and detection.
- Data set and modem control signals provided.
- Parity: (odd, even, none, mark, space).
- Full-duplex or half-duplex operation.
- 5, 6, 7, 8 and 9 bit transmission.

The SY6551 is an Asynchronous Communication Adapter (ACIA) intended to provide for interfacing the 6500/6800 microprocessor families to serial communication data sets and modems. A unique feature is the inclusion of an on-chip programmable baud rate generator, with a crystal being the only external component required.

PIN CONFIGURATION

Signal	Pin	6551	Pin	Signal
GND	1		28	R/$\overline{W}$
CS_0	2		27	φ2
$\overline{CS_1}$	3		26	$\overline{IRQ}$
$\overline{RES}$	4		25	DB_7
RxC	5		24	DB_6
XTAL1	6		23	DB_5
XTAL2	7		22	DB_4
$\overline{RTS}$	8		21	DB_3
$\overline{CTS}$	9		20	DB_2
TxD	10		19	DB_1
$\overline{DTR}$	11		18	DB_0
RxD	12		17	$\overline{DSR}$
RS_0	13		16	$\overline{DCD}$
RS_1	14		15	V_{CC}

Figure 1. Block Diagram

ORDERING INFORMATION

Part No.	Package	Clock Rate
SYC6551	Ceramic	1 MHz
SYD6551	Cerdip	1 MHz
SYP6551	Plastic	1 MHz
SYC6551A	Ceramic	2 MHz
SYD6551A	Ceramic	2 MHz
SYP6551A	Plastic	2 MHz

ABSOLUTE MAXIMUM RATINGS

Rating	Symbol	Allowable Range
Supply Voltage	V_{CC}	-0.3V to +7.0V
Input/Output Voltage	V_{IN}	-0.3V to +7.0V
Operating Temperature	T_{OP}	0°C to 70°C
Storage Temperature	T_{STG}	-55°C to 150°C

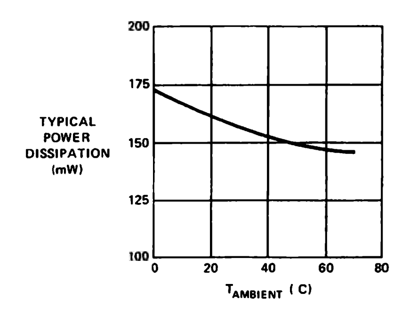
All inputs contain protection circuitry to prevent damage to high static charges. Care should be exercised to prevent unnecessary application of voltages in excess of the allowable limits.

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

D.C. CHARACTERISTICS (V_{CC} = 5.0V ± 5%, T_A = 0-70°C, unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage	V_{IH}	2.0	–	V_{CC}	V
Input Low Voltage	V_{IL}	-0.3	–	0.8	V
Input Leakage Current: V_{IN} = 0 to 5V (ϕ2, R/$\overline{W}$, $\overline{RES}$, CS_0, $\overline{CS_1}$, RS_0, RS_1, $\overline{CTS}$, RxD, $\overline{DCD}$, $\overline{DSR}$)	I_{IN}	–	±1.0	±2.5	μA
Input Leakage Current for High Impedance State (Three State)	I_{TSI}	–	±2.0	±10.0	μA
Output High Voltage: I_{LOAD} = −100μA (DB_0 - DB_7, TxD, RxC, $\overline{RTS}$, $\overline{DTR}$)	V_{OH}	2.4	–	–	V
Output Low Voltage: I_{LOAD} = 1.6mA (DB_0 - DB_7, TxD, RxC, $\overline{RTS}$, $\overline{DTR}$, $\overline{IRQ}$)	V_{OL}	–	–	0.4	V
Output High Current (Sourcing): V_{OH} = 2.4V (DB_0 - DB_7, TxD, RxC, $\overline{RTS}$, $\overline{DTR}$)	I_{OH}	-100	–	–	μA
Output Low Current (Sinking): V_{OL} = 0.4V (DB_0 - DB_7, TxD, RxC, $\overline{RTS}$, $\overline{DTR}$, $\overline{IRQ}$)	I_{OL}	1.6	–	–	mA
Output Leakage Current (Off State): V_{OUT} = 5V ($\overline{IRQ}$)	I_{OFF}	–	1.0	10.0	μA
Clock Capacitance (ϕ2)	C_{CLK}	–	–	20	pF
Input Capacitance (Except XTAL1 and XTAL2)	C_{IN}	–	–	10	pF
Output Capacitance	C_{OUT}	–	–	10	pF
Power Dissipation (See Graph) (T_A = 0°C) V_{CC} = 5.25V	P_D	–	170	300	mW

POWER DISSIPATION vs TEMPERATURE



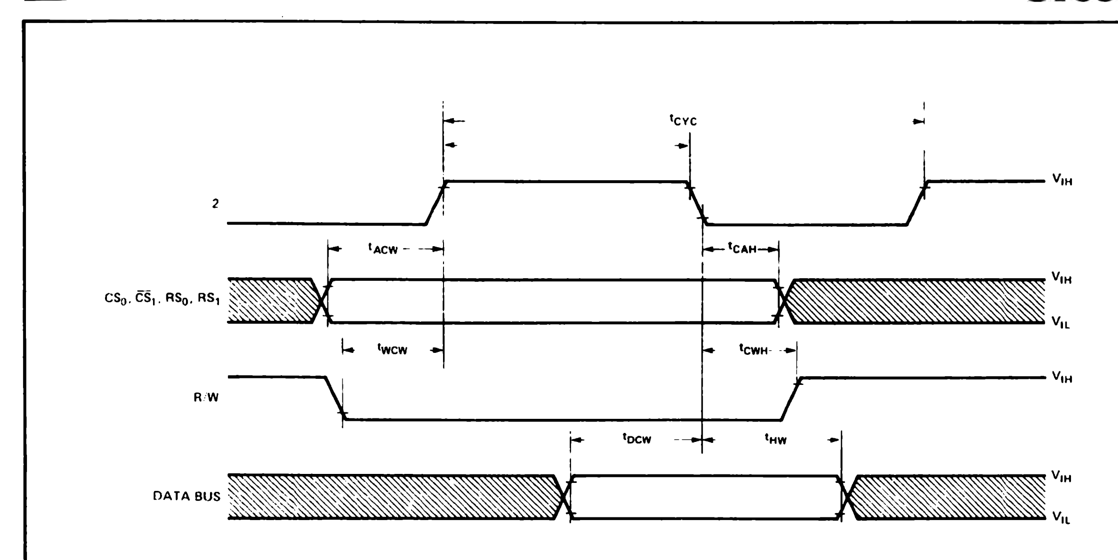


Figure 2. Write Timing Characteristics

WRITE CYCLE (V_{CC} = 5.0V ± 5%, T_A = 0 to 70°C, unless otherwise noted)

Characteristic	Symbol	SY6551 Min	SY6551 Max	SY6551A Min	SY6551A Max	Unit
Cycle Time	t_{CYC}	1.0	–	0.5		μs
ϕ2 Pulse Width	t_C	400	–	200	–	ns
Address Set-Up Time	t_{ACW}	120	–	70	–	ns
Address Hold Time	t_{CAH}	0	–	0	–	ns
R/$\overline{W}$ Set-Up Time	t_{WCW}	120	–	70	–	ns
R/$\overline{W}$ Hold Time	t_{CWH}	0	–	0	–	ns
Data Bus Set-Up Time	t_{DCW}	150	–	60	–	ns
Data Bus Hold Time	t_{HW}	20	–	20	–	ns

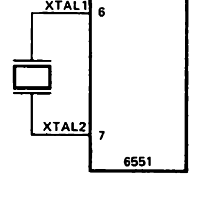
(t_r and t_f = 10 to 30 ns)

CRYSTAL SPECIFICATION

1. Temperature stability ± 0.01% (0° to 70°C)
2. Characteristics at 25°C ± 2°C
   - a. Frequency (MHz) 1.8432
   - b. Frequency tolerance (±%) 0.02
   - c. Resonance mode Series
   - d. Equivalent resistance (ohm) 400 max.
   - e. Drive level mW 2
   - f. Shunt capacitance pF 7 max.
   - g. Oscillation mode Fundamental

No other external components should be in the crystal circuit

CLOCK GENERATION



INTERNAL CLOCK

EXTERNAL CLOCK — XTAL1 6

OPEN CIRCUIT — XTAL2 7

EXTERNAL CLOCK

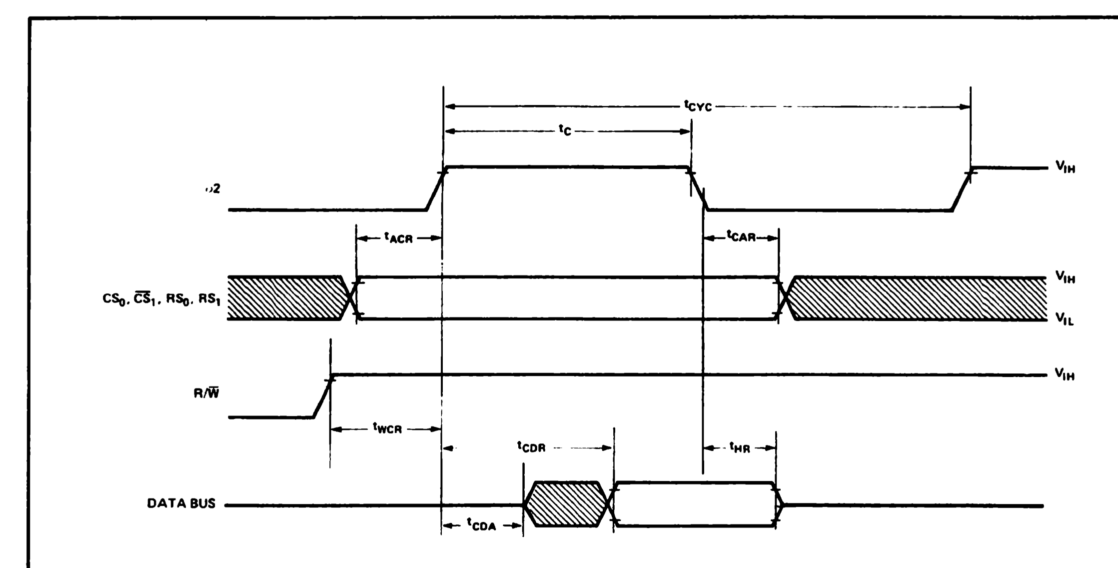
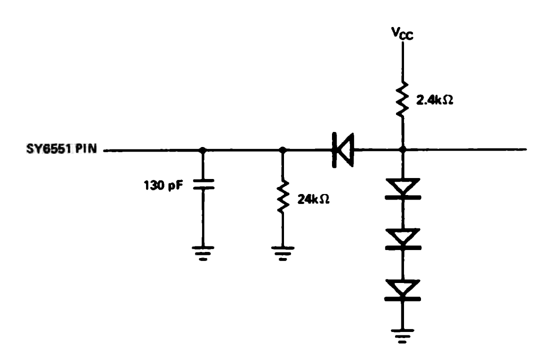


Figure 3. Read Timing Characteristics

READ CYCLE (V_{CC} = 5.0V ± 5%, T_A = 0 to 70°C, unless otherwise noted)

Characteristic	Symbol	SY6551		SY6551A		Unit
		Min	Max	Min	Max	
Cycle Time	t_{CYC}	1.0	–	0.5	–	μs
Pulse Width (φ2)	t_C	400	–	200	–	ns
Address Set-Up Time	t_{ACR}	120	–	70	–	ns
Address Hold Time	t_{CAR}	0	–	0	–	ns
R/$\overline{W}$ Set-Up Time	t_{WCR}	120	–	70	–	ns
Read Access Time (Valid Data)	t_{CDR}	–	200	–	150	ns
Read Data Hold Time	t_{HR}	20	–	20	–	ns
Bus Active Time (Invalid Data)	t_{CDA}	40	–	40	–	ns



TEST LOAD FOR DATA BUS (DB_0 – DB_7), $\overline{TxD}$, $\overline{DTR}$, RTS OUTPUTS

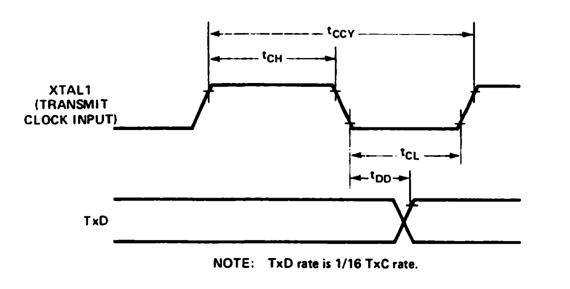


Figure 4a. Transmit Timing with External Clock

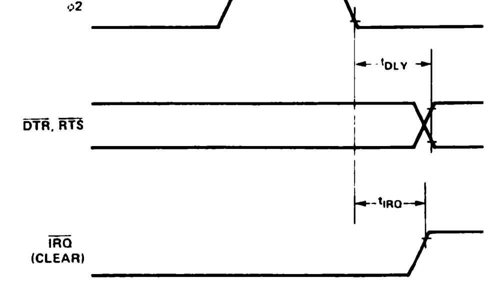


Figure 4b. Interrupt and Output Timing

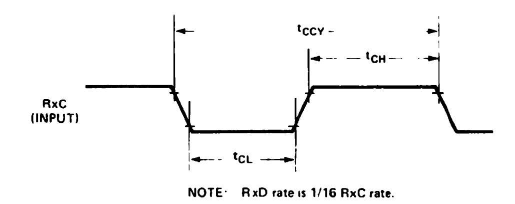


Figure 4c. Receive External Clock Timing

TRANSMIT/RECEIVE CHARACTERISTICS

Characteristic	Symbol	SY6551		SY6551A		Unit
		Min	Max	Min	Max	
Transmit/Receive Clock Rate	t_{CCY}	400*	–	400*	–	ns
Transmit/Receive Clock High Time	t_{CH}	175		175	–	ns
Transmit/Receive Clock Low Time	t_{CL}	175	–	175	–	ns
XTAL1 to TxD Propagation Delay	t_{DD}	–	500	–	500	ns
Propagation Delay ($\overline{RTS}$, $\overline{DTR}$)	t_{DLY}	–	500	–	500	ns
$\overline{IRQ}$ Propagation Delay (Clear)	t_{IRQ}	–	500	–	500	ns

(t_r, t_f = 10 to 30 nsec)

*The baud rate with external clocking is: Baud Rate = $\frac{1}{16 \times T_{CCY}}$

INTERFACE SIGNAL DESCRIPTION

$\overline{RES}$ (Reset)

During system initialization a low on the $\overline{RES}$ input will cause internal registers to be cleared.

$\phi 2$ (Input Clock)

The input clock is the system $\phi 2$ clock and is used to trigger all data transfers between the system microprocessor and the SY6551.

R/$\overline{W}$ (Read/Write)

The R/$\overline{W}$ is generated by the microprocessor and is used to control the direction of data transfers. A high on the R/$\overline{W}$ pin allows the processor to read the data supplied by the SY6551. A low on the R/$\overline{W}$ pin allows a write to the SY6551.

$\overline{IRQ}$ (Interrupt Request)

The $\overline{IRQ}$ pin is an interrupt signal from the interrupt control logic. It is an open drain output, permitting several devices to be connected to the common $\overline{IRQ}$ microprocessor input. Normally a high level, $\overline{IRQ}$ goes low when an interrupt occurs.

DB_0 - DB_7 (Data Bus)

The DB_0-DB_7 pins are the eight data lines used for transfer of data between the processor and the SY6551. These lines are bi-directional and are normally high-impedance except during Read cycles when selected.

CS_0, $\overline{CS_1}$ (Chip Selects)

The two chip select inputs are normally connected to the processor address lines either directly or through decoders. The SY6551 is selected when CS_0 is high and $\overline{CS_1}$ is low.

RS_ϕ, RS_1 (Register Selects)

The two register select lines are normally connected to the processor address lines to allow the processor to select the various SY6551 internal registers. The following table indicates the internal register select coding:

RS_1	RS_0	Write	Read
0	0	Transmit Data Register	Receiver Data Register
0	1	Programmed Reset (Data is "Don't Care")	Status Register
1	0	Command Register	
1	1	Control Register	

The table shows that only the Command and Control registers are read/write. The Programmed Reset operation does not cause any data transfer, but is used to clear the SY6551 registers. The Programmed Reset is slightly different from the Hardware Reset ($\overline{\text{RES}}$) and these differences are described in the individual register definitions.

ACIA/MODEM INTERFACE SIGNAL DESCRIPTION

XTAL1, XTAL2 (Crystal Pins)

These pins are normally directly connected to the external crystal (1.8432 MHz) used to derive the various baud rates. Alternatively, an externally generated clock may be used to drive the XTAL1 pin, in which case the XTAL2 pin must float.

TxD (Transmit Data)

The TxD output line is used to transfer serial NRZ (non-return-to-zero) data to the modem. The LSB (least significant bit) of the Transmit Data Register is the first data bit transmitted and the rate of data transmission is determined by the baud rate selected.

RxD (Receive Data)

The RxD input line is used to transfer serial NRZ data into the ACIA from the modem, LSB first. The receiver data rate is either the programmed baud rate or the rate of an externally generated receiver clock. This selection is made by programming the Control Register.

RxC (Receive Clock)

The RxC is a bi-directional pin which serves as either the receiver 16x clock input or the receiver 16x clock output. The latter mode results if the internal baud rate generator is selected for receiver data clocking.

$\overline{\text{RTS}}$ (Request to Send)

The $\overline{\text{RTS}}$ output pin is used to control the modem from the processor. The state of the $\overline{\text{RTS}}$ pin is determined by the contents of the Command Register.

$\overline{\text{CTS}}$ (Clear to Send)

The $\overline{\text{CTS}}$ input pin is used to control the transmitter operation. The enable state is with $\overline{\text{CTS}}$ low. The transmitter is automatically disabled if $\overline{\text{CTS}}$ is high.

$\overline{\text{DTR}}$ (Data Terminal Ready)

This output pin is used to indicate the status of the SY6551 to the modem. A low on $\overline{\text{DTR}}$ indicates the SY6551 is enabled and a high indicates it is disabled. The processor controls this pin via bit 0 of the Command Register.

$\overline{\text{DSR}}$ (Data Set Ready)

The $\overline{\text{DSR}}$ input pin is used to indicate to the SY6551 the status of the modem. A low indicates the "ready" state and a high, "not-ready." $\overline{\text{DSR}}$ is a high-impedance input and must not be a no-connect. If unused, it should be driven high or low, but not switched.

Note: If Command Register Bit 0 = 1 and a change of state on $\overline{\text{DSR}}$ occurs, $\overline{\text{IRQ}}$ will be set, and Status Register Bit 6 will reflect the new level. The state of $\overline{\text{DSR}}$ does not affect either Transmitter or Receiver operation.

$\overline{\text{DCD}}$ (Data Carrier Detect)

The $\overline{\text{DCD}}$ input pin is used to indicate to the SY6551 the status of the carrier-detect output of the modem. A low indicates that the modem carrier signal is present and a high, that it is not. $\overline{\text{DCD}}$, like $\overline{\text{DSR}}$, is a high-impedance input and must not be a no-connect.

Note: If Command Register Bit 0 = 1 and a change of state on $\overline{\text{DCD}}$ occurs, $\overline{\text{IRQ}}$ will be set, and Status Register Bit 5 will reflect the new level. The state of $\overline{\text{DCD}}$ does not affect Transmitter operation, but must be low for the Receiver to operate.

INTERNAL ORGANIZATION

The Transmitter/Receiver sections of the SY6551 are depicted by the block diagram in Figure 5.

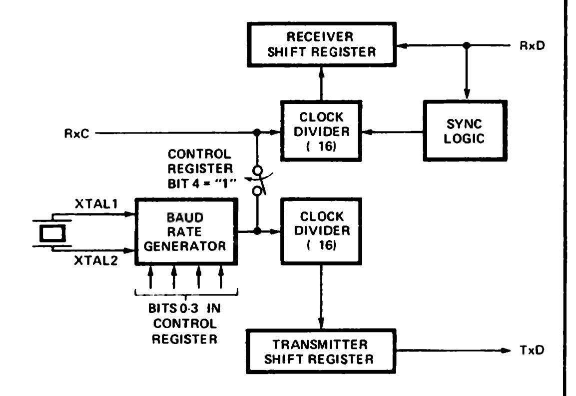


Figure 5. Transmitter/Receiver Clock Circuits

Bits 0-3 of the Control Register select the divisor used to generate the baud rate for the Transmitter. If the Receiver clock is to use the same baud rate as the Transmitter, then RxC becomes an output pin and can be used to slave other circuits to the SY6551.

CONTROL REGISTER

The Control Register is used to select the desired mode for the SY6551. The word length, number of stop bits, and clock controls are all determined by the Control Register, which is depicted in Figure 6.

CONTROL REGISTER

7	6	5	4	3	2	1	0

STOP BITS (bit 7)

0 = 1 Stop Bit
1 = 2 Stop Bits
1 Stop Bit if Word Length = 8 Bits and Parity*
1½ Stop Bits if Word Length = 5 Bits and No Parity.

WORD LENGTH (bits 6, 5)

BIT 6	BIT 5	DATA WORD LENGTH
0	0	8
0	1	7
1	0	6
1	1	5

RECEIVER CLOCK SOURCE (bit 4)

0 = External Receiver Clock
1 = Baud Rate Generator

BAUD RATE GENERATOR (bits 3–0)

3	2	1	0	BAUD RATE
0	0	0	0	16x EXTERNAL CLOCK
0	0	0	1	50 BAUD
0	0	1	0	75
0	0	1	1	109.92
0	1	0	0	134.58
0	1	0	1	150
0	1	1	0	300
0	1	1	1	600
1	0	0	0	1200
1	0	0	1	1800
1	0	1	0	2400
1	0	1	1	3600
1	1	0	0	4800
1	1	0	1	7200
1	1	1	0	9600
1	1	1	1	19,200

*This allows for 9-bit transmission (8 data bits plus parity)

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	0	0	0	0	0	0	0
PROGRAM RESET	–	–	–	–	–	–	–	–

Figure 6. Control Register Format

COMMAND REGISTER

The Command Register is used to control Specific Transmit/Receive functions and is shown in Figure 7.

COMMAND REGISTER

7	6	5	4	3	2	1	0

PARITY CHECK CONTROLS (bits 7, 6, 5)

BIT 7	BIT 6	BIT 5	OPERATION
–	–	0	Parity Disabled · No Parity Bit Generated · No Parity Bit Received
0	0	1	Odd Parity Receiver and Transmitter
0	1	1	Even Parity Receiver and Transmitter
1	0	1	Mark Parity Bit Transmitted, Parity Check Disabled
1	1	1	Space Parity Bit Transmitted, Parity Check Disabled

NORMAL/ECHO MODE FOR RECEIVER (bit 4)

0 = Normal
1 = Echo (Bits 2 and 3 must be "0")

DATA TERMINAL READY (bit 0)

0 = Disable Receiver and All Interrupts ($\overline{DTR}$ high)
1 = Enable Receiver and All Interrupts ($\overline{DTR}$ low)

RECEIVER INTERRUPT ENABLE (bit 1)

0 = $\overline{IRQ}$ Interrupt Enabled from Bit 3 of Status Register
1 = $\overline{IRQ}$ Interrupt Disabled

TRANSMITTER CONTROLS (bits 3, 2)

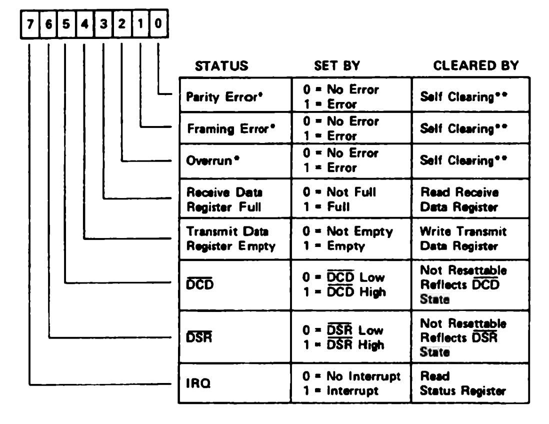
BIT 3	BIT 2	TRANSMIT INTERRUPT	$\overline{RTS}$ LEVEL	TRANSMITTER
0	0	Disabled	High	Off
0	1	Enabled	Low	On
1	0	Disabled	Low	On
1	1	Disabled	Low	Transmit BRK

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	0	0	0	0	0	0	0
PROGRAM RESET	–	–	–	0	0	0	0	0

Figure 7. Command Register Format

STATUS REGISTER

The Status Register is used to indicate to the processor the status of various SY6551 functions and is outlined in Figure 8.



Bit	STATUS	SET BY	CLEARED BY
0	Parity Error*	0 = No Error 1 = Error	Self Clearing**
1	Framing Error*	0 = No Error 1 = Error	Self Clearing**
2	Overrun*	0 = No Error 1 = Error	Self Clearing**
3	Receive Data Register Full	0 = Not Full 1 = Full	Read Receive Data Register
4	Transmit Data Register Empty	0 = Not Empty 1 = Empty	Write Transmit Data Register
5	$\overline{DCD}$	0 = $\overline{DCD}$ Low 1 = $\overline{DCD}$ High	Not Resettable Reflects $\overline{DCD}$ State
6	$\overline{DSR}$	0 = $\overline{DSR}$ Low 1 = $\overline{DSR}$ High	Not Resettable Reflects $\overline{DSR}$ State
7	IRQ	0 = No Interrupt 1 = Interrupt	Read Status Register

*NO INTERRUPT GENERATED FOR THESE CONDITIONS.
**CLEARED AUTOMATICALLY AFTER A READ OF RDR AND THE NEXT ERROR FREE RECEIPT OF DATA.

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	–	–	1	0	0	0	0
PROGRAM RESET	–	–	–	–	–	0	–	–

Figure 8. Status Register Format

TRANSMIT AND RECEIVE DATA REGISTERS

These registers are used as temporary data storage for the 6551 Transmit and Receive circuits. The Transmit Data Register is characterized as follows:

- Bit 0 is the leading bit to be transmitted.
- Unused data bits are the high-order bits and are "don't care" for transmission.

The Receive Data Register is characterized in a similar fashion:

- Bit 0 is the leading bit received.
- Unused data bits are the high-order bits and are "0" for the receiver.
- Parity bits are not contained in the Receive Data Register, but are stripped-off after being used for external parity checking. Parity and all unused high-order bits are "0".

Figure 9 illustrates a single transmitted or received data word, for the example of 8 data bits, parity, and 1 stop bit.

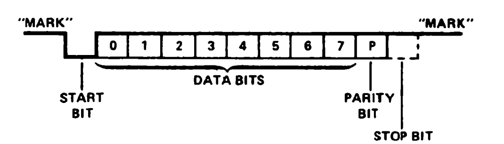
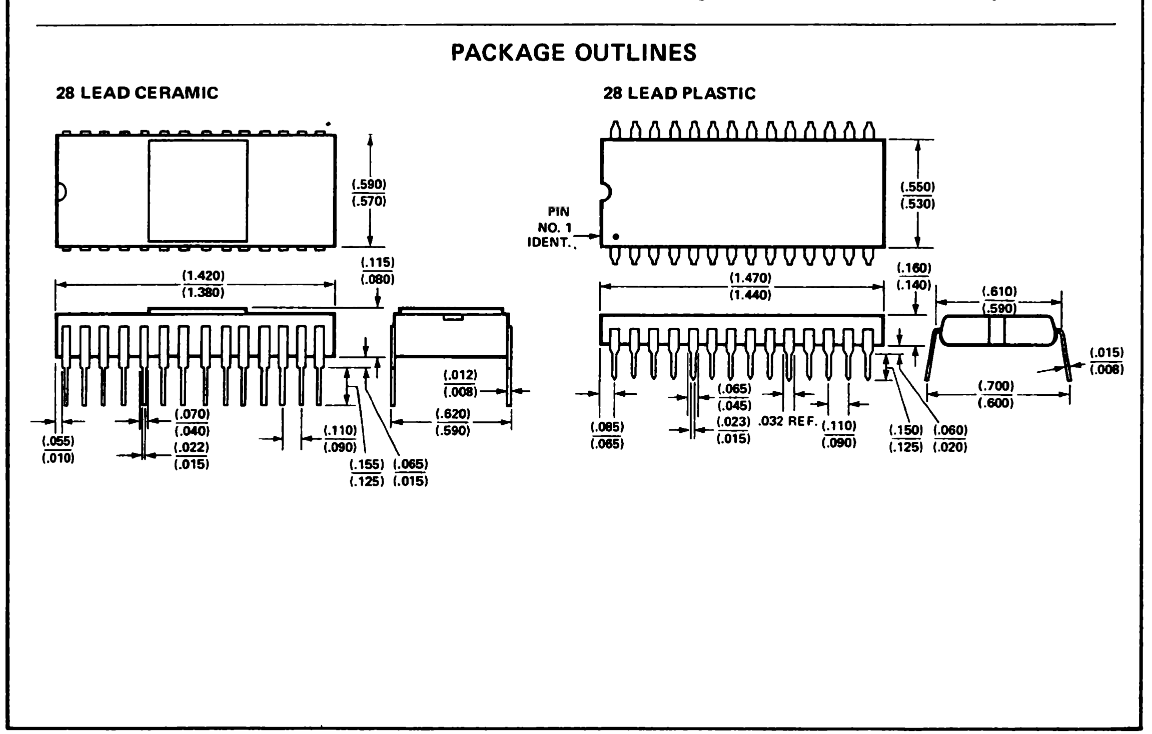


Figure 9. Serial Data Stream Example

PACKAGE OUTLINES



```
Current memory available:    9169
0000!
0000! 0001              chrfont .equ    1
0000!
0000!                           .include rom0
0000!                           .absolute
0000!                   ;
0000!                   ;       revision A3 Board
0000!                   ;
0000!                   ;
0000!                   ;  24*80 Monitor without tape IO
0000!                   ;
2 blocks for procedure code  8685 words left
```

```
0000:                           .proc       monitor
Current memory available:    8636
0000:
0000:                           .org    0F800
F800: 0000            LOC0     .equ    0
F800: 0001            LOC1     .equ    1
F800: 0020            WNDLFT   .equ    20
F800: 0021            WNDWDTH  .equ    21
F800: 0022            WNDTOP   .equ    22
F800: 0023            WNDBTM   .equ    23
F800: 0050            width    .equ    80.
F800: 0024            CH       .equ    24
F800: 0025            CV       .equ    25
F800: 0026            GBASL    .equ    26
F800: 0027            GBASH    .equ    27
F800: 0028            BASL     .equ    28
F800: 0029            BASH     .equ    29
F800: 002A            BAS2L    .equ    2A
F800: 002B            BAS2H    .equ    2B
F800: 002C            H2       .equ    2C
F800: 002C            LMNEM    .equ    2C
F800: 002D            V2       .equ    02D
F800: 002D            RMNEM    .equ    02D
F800: 002E            MASK     .equ    02E
F800: 002E            FORMAT   .equ 02E              ; dism
F800: 002F            LASTIN   .equ 02F              ; tape in
F800: 002F            LENGTH   .equ 02F              ; dism
F800: 0030            COLOR    .equ 030              ; LoRes color
F800: 0031            MODE     .equ 031              ; dism
F800: 0032            INVFLG   .equ 032              ; prompt char
F800: 0033            PROMPT   .equ 033
F800: 0034            YSAV     .equ 034
F800: 0035            YSAV1    .equ 035
F800: 0036            CSWL     .equ 036              ; output vector
F800: 0037            CSWH     .equ 037
F800: 0038            KSWL     .equ 038              ; input vector
F800: 0039            KSWH     .equ 039
F800: 003A            PCL      .equ 03A              ; go-, list-command
F800: 003B            PCH      .equ 03B
F800: 003C            A1L      .equ 03C
F800: 003D            A1H      .equ 03D
F800: 003E            A2L      .equ 03E
F800: 003F            A2H      .equ 03F
F800: 0040            A3L      .equ 040              ; memory set
F800: 0041            A3H      .equ 041
F800: 0042            A4L      .equ 042
F800: 0043            A4H      .equ 043
F800: 0045            ACC      .equ 45               ; 6502 register
F800: 0046            XREG     .equ ACC+1
F800: 0047            YREG     .equ ACC+2
F800: 0048            STATUS   .equ ACC+3
F800: 0049            SPNT     .equ ACC+4
F800: 004E            RNDL     .equ 04E              ; random number
F800: 004F            RNDH     .equ 04F
F800:
```

```
F800: 0200              IN      .equ 0200           ; Keyboard buffer
F800: 03F0              BRKV    .equ 03F0           ; brk vector
F800: 03F2              SOFTEV  .equ 03F2           ; soft reset vector
F800: 03F4              PWREDUP .equ 03F4
F800: 03F5              AMPERV  .equ 03F5           ; Applesoft &
F800: 03F8              USRADR  .equ 03F8           ; U-command
F800: 03FB              NMI     .equ 03FB           ; jmp  nmi
F800: 03FE              IRQLOC  .equ 03FE           ; jmp @irqloc
F800: 0400              LINE1   .equ 0400           ; first screen line
F800: 0479              chy     .equ    479         ; 80-col video driver
F800: 04F9              switch  .equ    4f9         ; 40/80-col switch
F800: 07F8              MSLOT   .equ    07F8        ; active slot ID
F800: C000              IOARD   .equ    0C000
F800: 00C0              iopage  .equ    0C0
F800: C000              KBD     .equ    0C000       ; ASCII input
F800: C008              Kbdextn .equ    0C008       ; functions Key input
F800: C006              chrBas  .equ    0C006       ; 64+64+128 set (inverse, flash, normal)
F800: C002              chrgen0 .equ    0C002       ; char gen A10
F800: C004              chrgen1 .equ    0C004       ; char gen A11
F800: C000              chrinv  .equ    0C000       ; invers/flash  switch
F800: C00A              vid40   .equ    0C00A       ; 40/80 col switch
F800: C00B              vid80   .equ    0C00B
F800: C00C              vidbnk  .equ    0C00C       ; video RAM switch
F800: C010              KBDSTRB .equ    0C010
F800: C020              TAPEOUT .equ    0C020
F800: C030              SPKR    .equ    0C030
F800: C050              TXTCLR  .equ    0C050
F800: C052              MIXCLR  .equ    0C052
F800: C054              LOWSCR  .equ    0C054
F800: C056              LORES   .equ    0C056
F800: C058              TTLout0 .equ    0C058     ; even: off, low  <= 0.4V
F800: C05A              TTLout1 .equ    0C05A     ; odd :  on, high >= 2.4 V
F800: C05C              TTLout2 .equ    0C05C
F800: C05E              TTLout3 .equ    0C05E
F800: C060              TAPEIN  .equ    0C060
F800: C064              PADDL0  .equ    0C064
F800: C070              PTRIG   .equ    0C070
F800: CFFF              CLRROM  .equ    0CFFF
F800: E000              BASIC   .equ    0E000
F800: E003              BASIC2  .equ    0E003
F800:
F800: 0080              bit7        .equ 80
F800:
F800:
F800:
F800:                           .include rom1
F800: 4A                PLOT    LSR     A
F801: 08                        PHP
F802: 20 ****                   JSR     GBASCALC
F805: 28                        PLP
F806: A9 0F                     LDA     #0F
F808: 90**                      BCC     $1
F80A: 69 E0                     ADC     #0E0
F80B* 02
F80C: 85 2E             $1      STA     MASK
```

```
F80E¦ 08                    PLOT1   php
F80F¦ 20 ****                       jsr     selbnk
F812¦ 4C ****                       jmp     plot80
F815¦ 00 00 00 00                   .org    0F819
F819¦ 20 00F8               HLINE   JSR     PLOT    ; Basic HLINE
F81C¦ C4 2C                 $1      CPY     H2
F81E¦ B0**                          BCS     RTS1
F820¦ C8                            INY
F821¦ 20 0EF8                       JSR     PLOT1
F824¦ 90F6                          BCC     $1
F826¦ 69 01                 VLINEZ  ADC     #1
F828¦ 48                    VLINE   PHA         ; Basic VLINE
F829¦ 20 00F8                       JSR     PLOT
F82C¦ 68                            PLA
F82D¦ C5 2D                         CMP     V2
F82F¦ 90F5                          BCC     VLINEZ
F81E* 11
F831¦ 60                    RTS1    RTS
F832¦
F832¦ A0 2F                 CLRSCR  LDY     #02F            ; Y-Max
F834¦ D0**                          BNE     CLRSC2
F836¦ A0 27                 CLRTOP  LDY     #27             ; Y-max, Basic GR
F834* 02
F838¦ 20 ****               CLRSC2  jsr     clrsc3
F83B¦ EA                            nop
F83C¦ A9 00                 $1      LDA     #0
F83E¦ 85 30                         STA     COLOR
F840¦ 20 28F8                       JSR     VLINE
F843¦ 88                            DEY
F844¦ 10F6                          BPL     $1
F846¦ 60                            RTS
F847¦
F847¦                               .org    0F847
F803* 47F8
F847¦ 48                    GBASCALC PHA
F848¦ 4A                            LSR     A
F849¦ 29 03                         AND     #3
F84B¦ 09 04                         ORA     #4      ; for LoRes Page 1
F84D¦ 85 27                         STA     GBASH
F84F¦ 68                            PLA
F850¦ 29 18                         AND     #18
F852¦ 90**                          BCC     $1
F854¦ 09 80                         ora     #80
F852* 02
F856¦ 85 26                 $1      STA     GBASL
F858¦ 0A                            ASL     A
F859¦ 0A                            ASL     A
F85A¦ 05 26                         ORA     GBASL
F85C¦ 85 26                         STA     GBASL
F85E¦ 60                            RTS
F85F¦
F85F¦ A5 30                 nxtcol  LDA     COLOR
F861¦ 18                            CLC
F862¦ 69 03                         ADC     #3
F864¦ 29 0F                 SETCOL  AND     #0F     ; Basic COLOR=
```

```
F866| 85 30                   STA    COLOR
F868| 0A                      ASL    A
F869| 0A                      ASL    A
F86A| 0A                      ASL    A
F86B| 0A                      ASL    A
F86C| 05 30                   ORA    COLOR
F86E| 85 30                   STA    COLOR
F870| 60                      RTS
F871|
F871| 4A              SCRN    LSR    A       ; Basic SCRN(X,Y) function
F872| 08                      PHP
F873| 20 ****                 jsr    scrn80
F876| EA                      nop
F877| EA                      nop
F878| 28                      PLP
F879| 90**            scrn2   BCC    $1
F87B| 4A                      LSR    A
F87C| 4A                      LSR    A
F87D| 4A                      LSR    A
F87E| 4A                      LSR    A
F879* 04
F87F| 29 0F           $1      AND    #0F
F881| 60                      RTS
F882|
```

```
F882¦                          .page
F882¦                          .ORG    0F882
F882¦ A6 3A           INSDS1   LDX     PCL
F884¦ A4 3B                    LDY     PCH
F886¦ 20 ****                  JSR     PRYX2
F889¦ 20 ****                  JSR     PRBLNK
F88C¦ A1 3A           INSDS2   LDA     @PCL,X
F88E¦ A8                       TAY
F88F¦ 4A                       LSR     A
F890¦ 90**                     BCC     IEVEN
F892¦ 6A                       ROR     A
F893¦ B0**                     BCS     ERR       ; all xxxxxx11 opcodes are illegal
F895¦ C9 A2                    CMP     #0A2      ; no STA # operation
F897¦ F0**                     BEQ     ERR
F899¦ 29 87                    AND     #87
F890* 09
F89B¦ 4A              IEVEN    LSR      A
F89C¦ AA                       TAX
F89D¦ BD ****                  LDA     FMT1,X
F8A0¦ 20 79F8                  JSR     SCRN2
F8A3¦ D0**                     BNE     GETFMT
F897* 0C
F893* 10
F8A5¦ A0 80           ERR      LDY     #80
F8A7¦ A9 00                    LDA     #0
F8A3* 04
F8A9¦ AA              GETFMT   TAX
F8AA¦ BD ****                  LDA     FMT2,X
F8AD¦ 85 2E                    STA     FORMAT
F8AF¦ 29 03                    AND     #3
F8B1¦ 85 2F                    STA     LENGTH
F8B3¦ 98                       TYA
F8B4¦ 29 8F                    AND     #8F
F8B6¦ AA                       TAX
F8B7¦ 98                       TYA
F8B8¦ A0 03                    LDY     #3
F8BA¦ E0 8A                    CPX     #8A
F8BC¦ F0**                     BEQ     MNNDX3
F8BE¦ 4A              MNNDX1   LSR     A
F8BF¦ 90**                     BCC     MNNDX3
F8C1¦ 4A                       LSR     A
F8C2¦ 4A              MNNDX2   LSR     A
F8C3¦ 09 20                    ORA     #20
F8C5¦ 88                       DEY
F8C6¦ D0FA                     BNE     MNNDX2
F8C8¦ C8                       INY
F8BF* 08
F8BC* 0B
F8C9¦ 88              MNNDX3   DEY
F8CA¦ D0F2                     BNE     MNNDX1
F8CC¦ 60                       RTS
```

```
F8CD:                              .page
F8CD: 00 00 00                     .org    0F8D0
F8D0: 20 82F8             INSTDSP  JSR     INSDS1
F8D3: 48                           PHA
F8D4: B1 3A               PRNTOP   LDA     @PCL, Y
F8D6: 20 ****                      JSR     PRBYTE
F8D9: A2 01                        LDX     #1
F8DB: 20 ****             PRNTBL   JSR     PRBL2
F8DE: C4 2F                        CPY     LENGTH
F8E0: C8                           INY
F8E1: 90F1                         BCC     PRNTOP
F8E3: A2 03                        LDX     #3
F8E5: C0 04                        CPY     #4
F8E7: 90F2                         BCC     PRNTBL
F8E9: 68                           PLA
F8EA: A8                           TAY
F8EB: B9 ****                      LDA     MNEML,Y ; print 3 characters, packed in 2 bytes
F8EE: 85 2C                        STA     LMNEM
F8F0: B9 ****                      LDA     MNEMR,Y
F8F3: 85 2D                        STA     RMNEM
F8F5:
F8F5: A9 00               $0       LDA     #0
F8F7: A0 05                        LDY     #5      ; shift 5 bits
F8F9: 06 2D               $1       ASL     RMNEM
F8FB: 26 2C                        ROL     LMNEM
F8FD: 2A                           ROL     A
F8FE: 88                           DEY
F8FF: D0F8                         BNE     $1
F901: 69 BF                        ADC     #0BF    ; "?"
F903: 20 ****                      JSR     COUT
F906: CA                           DEX
F907: D0EC                         BNE     $0
F909:
F909: 20 ****                      JSR     PRBLNK
F90C: A4 2F                        LDY     LENGTH
F90E: A2 06                        LDX     #6
F910: E0 03               PRADR1   CPX     #3
F912: F0**                         BEQ     PRADR5
F914: 06 2E               PRADR2   ASL     FORMAT
F916: 90**                         BCC     $0
F918: BD ****                      LDA     CHAR1-1,X
F91B: 20 ****                      JSR     COUT
F91E: BD ****                      LDA     CHAR2-1,X
F921: F0**                         BEQ     $0      ; no 2nd char
F923: 20 ****                      JSR     COUT
F921* 03
F916* 0E
F926: CA                  $0       DEX
F927: D0E7                         BNE     PRADR1  ; next format bit
F929: 60                           RTS
F92A: 88                  PRADR4   DEY
F92B: 30E7                         BMI     PRADR2
F92D: 20 ****                      JSR     PRBYTE
F912* 1C
```

```
F930: A5 2E          PRADR5  LDA   FORMAT
F932: C9 E8                  CMP   #0E8
F934: B1 3A                  LDA   @PCL,Y
F936: 90F2                   BCC   PRADR4
F938: 20 ****        RELADR  JSR   PCADJ3
F93B: AA                     TAX
F93C: E8                     INX
F93D: D0**                   BNE   PRNTYX
F93F: C8                     INY
F93D* 01
F940: 98             PRNTYX  TYA
F941: 20 ****        PRNTAX  JSR   PRBYTE
F944: 8A             PRNTX   TXA
F945: 4C ****                JMP   PRBYTE
F948:
F90A* 48F9
F88A* 48F9
F948: A2 03          PRBLNK  LDX   #3
F8DC* 4AF9
F94A: A9 A0          PRBL2   LDA   #0A0    ;' '
F94C: 20 ****        PRBL3   JSR   COUT
F94F: CA                     DEX
F950: D0F8                   BNE   PRBL2
F952: 60                     RTS
F953:
F953: 38             PCADJ   SEC
F954: A5 2F          PCADJ2  LDA   LENGTH
F939* 56F9
F956: A4 3B          PCADJ3  LDY   PCH
F958: AA                     TAX
F959: 10**                   BPL   PCADJ4
F95B: 88                     DEY
F959* 01
F95C: 65 3A          PCADJ4  ADC   PCL
F95E: 90**                   BCC   RTS2
F960: C8                     INY
F95E* 01
F961: 60             RTS2    RTS
F962:
```

```
F962|                          .page
F962|                    ; FMT1:  128 (dec) 4-bit pointer to the FMT2 table for xxxx xxx0 opcodes
F962|                    ;         16 (dec) 4-bit pointer to the FMT2 table for xxxx xx01 opcodes
F962|
F89E* 62F9
F962|                    FMT1
F962| 04 20 54 30 0D 80 04        .byte   004,020,054,030,00D,080,004,090
F969| 90
F96A| 03 22 54 33 0D 80 04        .byte   003,022,054,033,00D,080,004,090
F971| 90
F972| 04 20 54 33 0D 80 04        .byte   004,020,054,033,00D,080,004,090
F979| 90
F97A| 04 20 54 3B 0D 80 04        .byte   004,020,054,03B,00D,080,004,090
F981| 90
F982| 00 22 44 33 0D C8 44        .byte   000,022,044,033,00D,0C8,044,000
F989| 00
F98A| 11 22 44 33 0D C8 44        .byte   011,022,044,033,00D,0C8,044,0A9
F991| A9
F992| 01 22 44 33 0D 80 04        .byte   001,022,044,033,00D,080,004,090
F999| 90
F99A| 01 22 44 33 0D 80 04        .byte   001,022,044,033,00D,080,004,090
F9A1| 90
F9A2|                    ;  xxxx xx01 class:
F9A2| 26 31 87 9A                 .byte 026,031,087,09A   ; ORA,AND,EOR,ADC,STA,LDA,CMD,SBC
F9A6|
F9A6|                    ;  FMT2 bit 0..1 : instruction length-1
F9A6|                    ;  FMT2 bit 7..2 : if bit[i] then (print chr1[i-2],chr2[i-2])
F9A6|
F8AB* A6F9
F9A6| 00                 FMT2     .byte   00       ; illegal opcode
F9A7| 21                          .byte   21       ; #$hh
F9A8| 81                          .byte   81       ; $hh
F9A9| 82                          .byte   82       ; $dddd
F9AA| 00                          .byte   00       ;
F9AB| 00                          .byte   00       ;
F9AC| 59                          .byte   59       ; ($hh,X)
F9AD| 4D                          .byte   4D       ; ($hh),Y
F9AE| 91                          .byte   91       ; $hh,X
F9AF| 92                          .byte   92       ; $hhhh,X
F9B0| 86                          .byte   86       ; $hhhh,Y
F9B1| 4A                          .byte   4A       ; ($hhhh)
F9B2| 85                          .byte   85       ; $hh,Y
F9B3| 9D                          .byte   9D       ; $hhhh  special case: relative
F9B4|
F9B4|                             .org    0F9b4      ; char1/char2 used by mini assembler
F919* B3F9
F9B4| AC A9 AC A3 A8 A4   CHAR1   .byte 0AC,0A9,0AC,0A3,0A8,0A4       ; ",),#($"
F91F* B9F9
F9BA| D9 00 D8 A4 A4 00   CHAR2   .byte 0D9,000,0D8,0A4,0A4,000       ; "Y X$$ "
F9C0|
F8EC* C0F9
F9C0|                    MNEML
F9C0|                    ;
F9C0|                    ; IIIII000:
```

```
F9C0: 1C                .byte 01C      ; BRK
F9C1: 8A                .byte 08A      ; PHP
F9C2: 1C                .byte 01C      ; BPL
F9C3: 23                .byte 023      ; CLC
F9C4: 5D                .byte 05D      ; JSR
F9C5: 8B                .byte 08B      ; PLP
F9C6: 1B                .byte 01B      ; BMI
F9C7: A1                .byte 0A1      ; SEC
F9C8: 9D                .byte 09D      ; RTI
F9C9: 8A                .byte 08A      ; PHA
F9CA: 1D                .byte 01D      ; BVC
F9CB: 23                .byte 023      ; CLI
F9CC: 9D                .byte 09D      ; RTS
F9CD: 8B                .byte 08B      ; PLA
F9CE: 1D                .byte 01D      ; BVS
F9CF: A1                .byte 0A1      ; SEI
F9D0: 00                .byte 000      ; ?
F9D1: 29                .byte 029      ; DEY
F9D2: 19                .byte 019      ; BCC
F9D3: AE                .byte 0AE      ; TYA
F9D4: 69                .byte 069      ; LDY
F9D5: A8                .byte 0A8      ; TAY
F9D6: 19                .byte 019      ; BCS
F9D7: 23                .byte 023      ; CLV
F9D8: 24                .byte 024      ; CPY
F9D9: 53                .byte 053      ; IBY
F9DA: 1B                .byte 01B      ; BNE
F9DB: 23                .byte 023      ; CLD
F9DC: 24                .byte 024      ; CPX
F9DD: 53                .byte 053      ; INX
F9DE: 19                .byte 019      ; BEQ
F9DF: A1                .byte 0A1      ; SED
F9E0:
F9E0:           ; IIIxxx100:
F9E0: 00                .byte 000      ; ?
F9E1: 1A                .byte 01A      ; BIT
F9E2: 5B                .byte 05B      ; JMP
F9E3: 5B                .byte 05B      ; JMP
F9E4: A5                .byte 0A5      ; STY
F9E5: 69                .byte 069      ; LDY
F9E6: 24                .byte 024      ; CPY
F9E7: 24                .byte 024      ; CPX
F9E8:
F9E8:           ; IIII1010:
F9E8: AE                .byte 0AE      ; TXA
F9E9: AE                .byte 0AE      ; TXS
F9EA: A8                .byte 0A8      ; TAX
F9EB: AD                .byte 0AD      ; TSX
F9EC: 29                .byte 029      ; DEX
F9ED: 00                .byte 000      ; ?
F9EE: 7C                .byte 07C      ; NOP
F9EF: 00                .byte 000      ; ?
F9F0:
F9F0:           ; 0IIxxx10:
F9F0: 15                .byte 015      ; ASL
```

```
F9F1: 9C                        .byte 09C       ; ROL
F9F2: 6D                        .byte 06D       ; LSR
F9F3: 9C                        .byte 09C       ; ROR
F9F4:
F9F4:                     ; 1IIx0010, 1IIx0110, 1IIx1110:
F9F4: A5                        .byte 0A5       ; STX
F9F5: 69                        .byte 069       ; LDX
F9F6: 29                        .byte 029       ; DEC
F9F7: 53                        .byte 053       ; INC
F9F8:
F9F8:
F9F8:                     ; IIIxxx01:
F9F8: 84                        .byte 084       ; ORA
F9F9: 13                        .byte 013       ; AND
F9FA: 34                        .byte 034       ; EOR
F9FB: 11                        .byte 011       ; ADC
F9FC: A5                        .byte 0A5       ; STA
F9FD: 69                        .byte 069       ; LDA
F9FE: 23                        .byte 023       ; CMP
F9FF: A0                        .byte 0A0       ; SBC
FA00:
F8F1* 00FA
FA00: D8 62 5A 48 26 62 94 MNEMR  .byte   0D8, 062, 05A, 048, 026, 062, 094, 088
FA07: 88
FA08: 54 44 C8 54 68 44 E8       .byte   054, 044, 0C8, 054, 068, 044, 0E8, 094
FA0F: 94
FA10: 00 B4 08 84 74 B4 28       .byte   000, 0B4, 008, 084, 074, 0B4, 028, 06E
FA17: 6E
FA18: 74 F4 CC 4A 72 F2 A4       .byte   074, 0F4, 0CC, 04A, 072, 0F2, 0A4, 08A
FA1F: 8A
FA20:
FA20: 00 AA A2 A2 74 74 74       .byte   000, 0aa, 0a2, 0a2, 074, 074, 074, 72
FA27: 72
FA28:
FA28: 44 68 B2 32 B2 00 22       .byte   044,  68, 0b2,  32, 0b2,   0,  22,  0
FA2F: 00
FA30:
FA30: 1A 1A 26 26                .byte   1a, 1a, 26, 26
FA34: 72 72 88 C8                .byte   72, 72, 88,0c8
FA38:
FA38: C4 CA 26 48 44 44 A2       .byte   0c4,0ca,026,048,044,044,0a2,0c8
FA3F: C8
FA40:
FA40:
FA40:
FA40:
FA40:                           .include rom2
```

```
FA40:                  .page          ; filer ROM2.text
FA40:                  .org    0FA40
FA40: 85 45     irq    sta     acc
FA42: 68               pla
FA43: 48               pha
FA44: 29 10            and     #10    ; test break flag, bit 4
FA46: D0**             bne     break
FA48: 6C FE03          jmp     @irqloc
FA4B:
FA4B: 00               .org    0FA4C
FA46* 04
FA4C: 28        break  plp
FA4D: 20 ****          jsr     savl
FA50: 68               pla
FA51: 85 3A            sta     pcl
FA53: 68               pla
FA54: 85 3B            sta     pch
FA56: 6C F003          jmp     @brkv
FA59:
FA59: 20 82F8   oldbrk jsr     insdsl
FA5C: 20 ****          jsr     rgdspl
FA5F: 4C ****          jmp     mon
FA62:
FA62:                  .org    0FA62
FA62: D8        reset  cld
FA63: 20 ****          jsr     setnorm
FA66: 20 ****          jsr     init
FA69: 20 ****          jsr     setvid
FA6C: 20 ****          jsr     setkbd
FA6F:
FA6F: D8        newmon cld
FA70: 20 ****          jsr     bell
FA73:                  .if     chrfont=1      ; national
FA73: 8D 03C0          sta     chrgen0+1
FA76: 8D 04C0          sta     chrgen1
FA79:                  .endc
FA79:                  .if     chrfont=2      ; ASCII
FA79:                  .endc
FA79:                  .if      chrfont=3     ; APL
FA79:                  .endc
FA79:
FA79: 8D 00C0          sta     chrinv
FA7C: 2C FFCF          bit     clrROM
FA7F: 2C 10C0          bit     kbdstrb
FA82: AD F303          lda     softev+1
FA85: 49 A5            eor     #0a5
FA87: CD F403          cmp     pwredup
FA8A: D0**             bne     pwrup
FA8C: AD F203          lda     softev
FA8F: D0**             bne     nofix
FA91: A9 E0            lda     #0e0
FA93: CD F303          cmp     softev+1
FA96: D0**             bne     nofix
FA98:
```

```
FA98: A0 03              fixsev  ldy   #3
FA9A: 8C F203                    STY   SOFTEV
FA9D: 4C 00E0                    JMP   BASIC
FA96* 08
FA8F* 0F
FAA0: 6C F203            NOFIX   JMP   @SOFTEV
FAA3:
FA8A* 17
FAA3: 38                 PWRUP   sec
FAA4: 6E F904                    ror   switch
FAA7: 20 ****                    JSR   LOGO1
FAAA: A2 05              SETPG3  LDX   #5
FAAC: BD ****            SETPLP  LDA   PWRCON-1,X
FAAF: 9D EF03                    STA   BRKV-1,X
FAB2: CA                         DEX
FAB3: D0F7                       BNE   SETPLP
FAB5: A9 C8                      LDA   #0C8    ; last slot+1
FAB7: 85 01                      STA   LOC1    ; SET PTR H
FAB9: 86 00                      STX   LOC0    ; Xreg=0
FABB: A0 07              SLOOP   LDY   #7      ; Y is byte offset into the slot ROM
FABD: C6 01                      DEC   LOC1
FABF: A5 01                      LDA   LOC1
FAC1: C9 C1                      CMP   #0C1    ; slot=1?
FAC3: F0D3                       BEQ   FIXSEV  ; yes, slot 1 is the builtin printer
FAC5: 8D F807                    STA   MSLOT
FAC8: B1 00              $0      LDA   @LOC0,Y ; read slot ROM
FACA: D9 ****                    CMP   DISKID,Y        ; is it a boot device (floppy, harddisk...) ??
FACD: D0EC                       BNE   SLOOP   ; no, test next slot
FACF: 88                         DEY
FAD0: 88                         DEY           ; yes so check next odd byte
FAD1: 10F5                       BPL   $0
FAD3: 6C 0000                    jmp   @LOC0   ; it is a disk! jump to boot
```

```
FAD6:                          .page
FAD6: 00                       .ORG    0FAD7
FAD7: 20 ****           REGDSP JSR     CROUT
FA5D* DAFA
FADA: A9 45             RGDSP1 LDA     #ACC
FADC: 85 40                    STA     A3L
FADE: A9 00                    LDA     #0
FAE0: 85 41                    STA     A3H
FAE2: A2 FB                    LDX     #0FB    ; -5
FAE4: A9 A0             $1     LDA     #0A0
FAE6: 20 ****                  JSR     COUT
FAE9: BD ****                  LDA     RTBL-251., X
FAEC: 20 ****                  JSR     COUT
FAEF: A9 BD                    LDA     #0BD    ; "="
FAF1: 20 ****                  JSR     COUT
FAF4: B5 4A                    LDA     ACC+5, X
FAF6: 20 ****                  JSR     PRBYTE
FAF9: E8                       INX
FAFA: 30E8                     BMI     $1
FAFC: 60                       RTS
FAFD:
FAFD:
FAAD* FCFA
FAFD: 59FA              pwrcon .word   OLDBRK
FAFF: 00E0                     .word   basic
FB01:
FACB* 01FB
FB01: 45 20             diskid eor     20      ; opcode (0E0^0A5=45) used for mask!!
FB03: A0 00                    ldy     #0      ; code never executed,
FB05: A2 03                    ldx     #3      ; only for disk ID
FB07: 86 3C                    stx     3c
FB09:
FB09: 08 15 0A 0B 40 0E 0F locchr .byte 08,15,0a,0b,40,0e,0f
FB10:
FB10: D0**              sw1    bne     sw2
FB12: 0A                       asl     a
FB10* 01
FB13: 8D 7904           sw2    sta     chy
FB16: 4C ****                  jmp     scrl00
FB19:
FB19:                          .ORG    0FB19
FAE4* 1EFA
FB19: C1 D8 D9 D0 D3    RTBL   .byte   0C1,0D8,0D9,0D0,0D3   ; "AXYPS"
FB1E:
FB1E: AD 70C0           PREAD  LDA     PTRIG   ; Basic PDL(n) function
FB21: A0 00                    LDY     #0
FB23: EA                       NOP
FB24: EA                       NOP
FB25: BD 64C0           $1     LDA     PADDL0,X
FB28: 10**                     BPL     $2
FB2A: C8                       INY
FB2B: D0F8                     BNE     $1
FB2D: 88                       DEY
FB28* 04
```

```
FB2E: 60                  $2      RTS
FB2F:
FA67* 2FFB
FB2F: A9 04               INIT    LDA     #4      ; set I flag!
FB31: 85 48                       STA     STATUS
FB33: 2C 56C0                     BIT     LORES
FB36: 2C 54C0                     bit     lowscr
FB39: 2C 51C0             SETTXT  BIT     TxtClr+1        ; set text mode, Basic TEXT
FB3C: A9 00                       LDA     #0
FB3E: F0**                        BEQ     SETWND
FB40: 2C 50C0             SETGR   BIT     TXTCLR          ; set graphic,  Basic GR
FB43: 2C 53C0                     BIT     MIXclr+1        ; set mixed mode
FB46: 20 36F8                     JSR     CLRTOP
FB49: A9 14                       LDA     #14
FB3E* 0B
FB4B: 85 22               SETWND  STA     WNDTOP
FB4D: A9 00                       lda     #0
FB4F: 85 20                       sta     wndlft
FB51: A9 50                       LDA     #width
FB53: 85 21                       STA     WNDWDTH
FB55: A9 18                       LDA     #18
FB57: 85 23                       STA     WNDBTM
FB59: A9 17                       LDA     #17
FB5B: 85 25               TABV    STA     CV
FB5D: 4C ****                     JMP     VTAB
FB60:
FB60: 20 ****             LOGO    JSR     HOME      ;   CLEAR THE SCRN
FB63: A0 08                       LDY     #8
FB65: B9 ****             $1      LDA     TITLE, Y  ;   GET A CHAR
FB68: 99 0004                     STA     LINE1, Y
FB6B: 88                          DEY
FB6C: 10F7                        bpl     $1
FB6E: 60                          RTS
FB6F:
FB6F:                             .org    0FB6F
FB6F: AD F303             SETPWRC LDA     SOFTEV+1
FB72: 49 A5                       EOR     #0A5
FB74: 8D F403                     STA     PWREDUP
FB77: 60                          RTS
FB78:
FB78: AC 00C0             VIDWAIT LDY     KBD
FB7B: C0 93                       CPY     #93      ; ctrl-S pressed?
FB7D: D0**                        BNE     $2       ; no so continue
FB7F: 2C 10C0                     BIT     KBDSTRB  ; clear keyboard strobe
FB82: AC 00C0             $1      LDY     KBD      ; wait until next key pressed
FB85: 10FB                        BPL     $1
FB87: C0 83                       CPY     #83      ; ctrl-C?
FB89: F0**                        BEQ     vidout   ; yes, it is for Basic
FB8B: 8D 10C0                     sta     KBDSTRB  ; clear strobe
FB7D* 0F
FB8E: D0**                $2      bne     VIDOUT   ; display char in accu
FB90:
FB66* 90FB
FB90: C2 E1 F3 E9 F3 A0 B1  TITLE   .byte   0C2,0E1,0F3,0E9,0F3,0A0,0B1,0B0,0B8    ; "Basis 108"
FB97: B0 B8
```

```
FB99:
FB99: 0F 3E 65 19 57 9B 41  locjmp  .byte   0F,3E,65,19,57,9b,41
FBA0: A0 07                 local   ldy     #7
FBA2: D9 09FB               $1      cmp     locchr,y
FBA5: D0**                          bne     $2
FBA7: A9 FC                         lda     #0fc
FBA9: 48                            pha
FBAA: B9 99FB                       lda     locjmp,y
FBAD: 48                            pha
FBAE: A0 18                         ldy     #18
FBB0: D0**                          bne     pip      ; echo for legel keys
FBA5* 0B
FBB2: 88                    $2      dey
FBB3: 10ED                          bpl     $1
FBB5: 60                            rts
FBB6: 20 A0FB               jlocal  jsr     local
FBB9: 20 ****               rdchar1 jsr     rdkey
FBBC: 29 FF                         and     #0ff     ; test bit 7
FBBE: 10F6                          bpl     jlocal
FBC0: 60                            rts
FBC1:
FBC1:                               .ORG    0FBC1
FBC1: 48                    BASCALC PHA
FBC2: 4A                            LSR     A
FBC3: 29 03                         AND     #3
FBC5: 09 04                         ORA     #4       ; for text page 1
FBC7: 85 29                         STA     BASH
FBC9: 68                            PLA
FBCA: 29 18                         AND     #18
FBCC: 90**                          BCC     $1
FBCE: 09 80                         ora     #80
FBCC* 02
FBD0: 85 28                 $1      STA     BASL
FBD2: 0A                            ASL     A
FBD3: 0A                            ASL     A
FBD4: 05 28                         ORA     BASL
FBD6: 85 28                         STA     BASL
FBD8: 60                            RTS
FBD9:
FBD9: C9 87                 BELL1   CMP     #87
FBDB: D0**                          BNE     noctrl
FBDD: A0 70                         LDY     #070     ; new sound
FBB0* 2D
FBDF: 98                    pip     tya              ; another sound
FBE0: 4A                            lsr     A
FBE1: 4A                            lsr     A
FBE2: 09 07                         ora     #7       ; set minimum time
FBE4: 20 ****                       jsr     WAIT
FBE7: 2C 30C0                       bit     SPKR
FBEA: 88                            dey
FBEB: D0F2                          bne     pip
FBDB* 10
FBED: 60                    noctrl  rts
FBEE:
FBEE:                               .org    0FBEE
```

```
FBEE: 25 32            storinv and     invflg
FBF0: 20 ****          STORADV jsr     stor80
FBF3: EA                       nop
FBF4: E6 24            ADVANCE INC     CH
FBF6: A4 24                    ldy     CH
FBF8: C4 21                    cpy     WNDWDTH
FBFA: B0**                     BCS     CR
FBFC: 60                       RTS
FBFD:
FBFD:                          .org    0FBFD
FB8E* 6D
FB89* 72
FBFD: C9 A0            VIDOUT  cmp     #0A0    ; ctrl?
FBFF: B0ED                     bcs     storinv ; no, diplay it normal or inverse
FC01: A8                       tay
FC02: 10EC                     bpl     STORADV
FC04: C9 8D                    CMP     #8D
FC06: F0**                     BEQ     CR
FC08: C9 8A                    CMP     #8A
FC0A: F0**                     BEQ     LF
FC0C: C9 88                    CMP     #88
FC0E: D0C9                     BNE     BELL1
FC10: C6 24            BS      DEC     CH
FC12: 10**                     BPL     RTS4
FC14: A5 21                    LDA     WNDWDTH
FC16: 85 24                    STA     CH
FC18: C6 24                    DEC     CH
FC1A: A5 22            UP      LDA     WNDTOP
FC1C: C5 25                    CMP     CV
FC1E: B0**                     BCS     RTS4
FC20: C6 25                    DEC     CV
FB5E* 22FC
FC22: A5 25            VTAB    LDA     CV
FC24: 20 C1FB          VTABZ   JSR     BASCALC
FC27: 4C ****                  jmp     vtab80
FC1E* 0A
FC12* 16
FC2A: 60               RTS4    RTS
FC2B:
FC2B: B9 0002          getupcs lda     in,y    ;       read uppercase char from input buffer
FC2E: C8                       iny
FC2F: C9 E0            upper   cmp     #0E0    ;
FC31: 90**                     bcc     $1
FC33: 29 DF                    and     #0DF    ;       shift to uppercase
FC31* 02
FC35: 60               $1      rts
FC36:
FC36: 48               sw5     pha
FC37: 98               sw6     tya
FC38: 4A                       lsr     a
FC39: 8D 0BC0                  sta     vid80
FC3C: 4C ****                  jmp     selbnk2
FC3F:
FC3F:                          .org    0FC3F
FC3F: 4C F4FB                  jmp     advance ;       cursor right jmp
```

```
FC42¦                        .ORG    0FC42
FC42¦ A4 24          CLREOP  LDY     CH
FC44¦ A5 25                  LDA     CV
FC46¦ 48             CLEOP1  PHA
FC47¦ 20 24FC                JSR     VTABZ
FC4A¦ 20 ****                JSR     CLEOLZ
FC4D¦ A0 00                  LDY     #0
FC4F¦ 68                     PLA
FC50¦ 69 00                  ADC     #0      ; carry=1 from cleolz
FC52¦ C5 23                  CMP     WNDBTM
FC54¦ 90F0                   BCC     CLEOP1
FC56¦ B0CA                   BCS     VTAB
FB61* 58FC
FC58¦ A5 22          HOME    LDA     WNDTOP
FC5A¦ 85 25                  STA     CV
FC5C¦ A0 00                  LDY     #0
FC5E¦ 84 24                  STY     CH
FC60¦ F0E4                   BEQ     CLEOP1
FC62¦
```

```
FC62:                   .page
FC06* 5A
FBFA* 66
FC62: A9 00    CR       LDA     #0
FC64: 85 24             STA     CH
FC0A* 5A
FC66: E6 25    LF       INC     CV
FC68: A5 25             lda     cv
FC6A: C5 23             cmp     wndbtm
FC6C: 90B6              bcc     vtabz
FC6E: C6 25             dec     cv
FC70: A5 22    scroll   lda     wndtop
FC72: 48                PHA
FC73: 20 24FC           JSR     VTABZ
FC76: A5 28    $1       LDA     BASL
FC78: 85 2A             STA     BAS2L
FC7A: A5 29             LDA     BASH
FC7C: 85 2B             STA     BAS2H
FC7E: A4 21             LDY     WNDWDTH
FC80: 88                DEY
FC81: 68                PLA
FC82: 69 01             ADC     #1      ; carry=0 from scroll line
FC84: C5 23             CMP     WNDBTM
FC86: B0**              BCS     $3
FC88: 48                PHA
FC89: 20 24FC           JSR     VTABZ
FC8C: 98                tya
FC8D: AC F904           ldy     switch
FC90: 20 10FB           jsr     sw1     ; on return carry=0
FC93: 90E1              bcc     $1      ; bra $1
FC95:
FC95:                   .org    0FC95
FC86* 0D
FC95: A0 00    $3       LDY     #0
FC97: 20 ****           JSR     CLEOLZ
FC9A: B086              BCS     VTAB
FC9C: A4 24    CLREOL   LDY     CH
FC98* 9EFC
FC4B* 9EFC
FC9E: 38       CLEOLZ   sec     ; carry=1 after plp
FC9F: 08                php
FCA0: 4C ****           jmp     cleol80
```

```
FCA3:                          .page
FCA3: 00 00 00 00 00           .org    0FCA8
FBE5* A8FC
FCA8: 38               WAIT    SEC             ; wait for ord(Accu^2) time
FCA9: 48               $1      PHA
FCAA: E9 01            $2      SBC     #1
FCAC: D0FC                     BNE     $2
FCAE: 68                       PLA
FCAF: E9 01                    SBC     #1
FCB1: D0F6                     BNE     $1
FCB3: 60                       RTS
FCB4:
FCB4: E6 42            NXTA4   INC     A4L
FCB6: D0**                     BNE     NXTA1
FCB8: E6 43                    INC     A4H
FCB6* 02
FCBA: A5 3C            NXTA1   LDA     A1L
FCBC: C5 3E                    CMP     A2L
FCBE: A5 3D                    LDA     A1H
FCC0: E5 3F                    SBC     A2H
FCC2: E6 3C                    INC     A1L
FCC4: D0**                     BNE     $2
FCC6: E6 3D                    INC     A1H
FCC4* 02
FCC8: 60               $2      RTS
```

```
FCC9:                         .page
FCC9:                   ;
FCC9:                   ;     80-col screen driver
FCC9:                   ;
FCC9: 4C ****           selbnk  jmp     sw3
FC3D* CCFC
FCCC: 8D 0CC0           selbnk2 sta     vidbnk  ; 400..BFF: dynamic RAM
FCCF: 90**                      bcc     $1
FCD1: 78                        sei
FCD2: 8D 0DC0                   sta     vidbnk+1; 400..BFF: static RAM
FCCF* 04
FCD5: 8C 7904           $1      sty     chy     ; save  Yreg in active bank!
FCD8: A8                        tay             ; for   lda/sta @basl,y
FCD9: 68                        pla
FCDA: 60                        rts
FCDB:
FCA1* DBFC
FCDB: 20 C9FC           cleol80 jsr     selbnk          ; clear to end of line
FCDE: A9 A0                     lda     #0A0
FCE0: 91 28                     sta     @basl,Y
FCE2: AC 7904                   ldy     chy
FCE5: C8                        iny
FCE6: C4 21                     cpy     wndwdth
FCE8: 90F1                      bcc     cleol80
FCEA: 4C ****                   jmp     vidplp
FCED:
FCED: B1 26             plot80  lda     @gbasl,y        ; MiRes plot
FCEF: 45 30                     eor     color
FCF1: 25 2E                     and     mask
FCF3: 51 26                     eor     @gbasl,y
FCF5: 91 26                     sta     @gbasl,y
FCF7: 4C ****                   jmp     vidrts
FCFA:
FCCA* FAFC
FCFA: 48                sw3     pha
FCFB: AD F904                   lda     switch
FCFE: F0**                      beq     sw4
FD00: 4C 37FC                   jmp     sw6
FCFE* 03
FD03: 68                sw4     pla
FD04: 8C 7904                   sty     chy
FD07: 8D 0AC0                   sta     vid40
FD0A: 60                        rts
FD0B:
```

```
FD0B:                       .page
FD0B: 00                    .org   0FD0C
FBBA* 0CFD
FD0C: 4C ****       RDKEY   jmp    rdKey2
FAA8* 0FFD
FD0F: 20 2FFB       logo1   jsr    init
FD12: 4C 60FB               jmp    logo
FD0D* 15FD
FD15: 20 ****       rdKey2  jsr    curs80
FD18:                       .org   0FD18
FD18: 6C 3800               jmp    @kswl
FD1B:
FD1B:                       .org   0FD1B
FD1B: E6 4E         KEYIN   INC    RNDL    ; slow human is the random generator
FD1D: D0**                  BNE    $1
FD1F: E6 4F                 INC    RNDH
FD1D* 02
FD21: 2C 00C0       $1      BIT    KBD     ; Key pressed?
FD24: 10F5                  BPL    KEYIN
FD26: 20 ****               jsr    curs80  ; remove cursor
FD29: AD 08C0               lda    Kbdextn ; read function Key bit
FD2C: 29 80                 and    #bit7
FD2E: 4D 00C0               eor    KBD     ; merge with ASCII code
FD31: 8D 10C0               sta    KBDSTRB
FD34: 60                    rts
FD35:
FD35:                       .org   0FD35
FD35: 4C B9FB       RDCHAR  jmp    rdchar1
FD38:
```

```
FD38¦                           .page
FD38¦ 00 00 00 00 00            .org    0FD3D
FD3D¦ A5 32            NOTCR    LDA     INVFLG
FD3F¦ 48                        PHA
FD40¦ A9 FF                     LDA     #0FF
FD42¦ 85 32                     STA     INVFLG
FD44¦ BD 0002                   LDA     IN,X
FD47¦ 20 ****                   JSR     COUT
FD4A¦ 68                        PLA
FD4B¦ 85 32                     STA     INVFLG
FD4D¦ BD 0002                   LDA     IN,X
FD50¦ C9 88                     CMP     #088    ; ctrl-H
FD52¦ F0**                      BEQ     BCKSPC
FD54¦ C9 98                     CMP     #098    ; ctrl-X
FD56¦ F0**                      BEQ     CANCEL
FD58¦ E0 F8                     CPX     #0F8
FD5A¦ 90**                      BCC     NOTCR1
FD5C¦ 20 ****                   JSR     BELL
FD5A* 03
FD5F¦ E8               NOTCR1   INX
FD60¦ D0**                      BNE     NXTCHAR
FD56* 0A
FD62¦ A9 A3            CANCEL   LDA     #0A3    ; "#" like MBasic 5.2
FD64¦ 20 ****                   JSR     COUT
FD67¦ 20 ****          GETLNZ   JSR     CROUT
FD6A¦ A5 33            GETLN    LDA     PROMPT
FD6C¦ 20 ****                   JSR     COUT
FD6F¦ A2 01                     LDX     #1
FD52* 1D
FD71¦ 8A               BCKSPC   TXA
FD72¦ F0F3                      BEQ     GETLNZ
FD74¦ CA                        DEX
FD60* 13
FD75¦ 20 35FD          NXTCHAR  JSR     RDCHAR
FD78¦ C9 95                     CMP     #95     ; ctrl-U
FD7A¦ D0**                      BNE     ADDINP
FD7C¦ 20 ****                   jsr     get80
FD7F¦ EA                        nop
FD80¦ EA                        nop
FD81¦ EA                        nop
FD82¦ EA                        nop
FD83¦ EA                        nop
FD84¦                           .org    0FD84
FD7A* 08
FD84¦ 9D 0002          ADDINP   STA     IN, X
FD87¦ C9 8D                     CMP     #8D
FD89¦ D0B2                      BNE     NOTCR
FD8B¦                           .ORG    0FD8B
FD8B¦ 20 9CFC          $1       JSR     CLREOL  ; entry by DOS 3.3 toolkit asmb!
FD68* 8EFD
FAD8* 8EFD
FD8E¦ A9 8D            CROUT    lda     #8d
FD90¦ D0**                      BNE     COUT
FD92¦
```

```
FD92!                    .page
FD92!                    .org    0FD92
FD92! A4 3D      pral    ldy     alh
FD94! A6 3C              ldx     all
FD96! 20 ****    pryx2   jsr     newln
FD99! 20 40F9            jsr     prntyx
FD9C! A0 00              ldy     #0
FD9E! A9 BA              lda     #0BA    ; ':'
FDA0! 4C ****            jmp     cout
FDA3!
FDA3! A5 3C      XAM8    LDA     A1L
FDA5! 09 0F              ora     #0f
FDA7! 85 3E              STA     A2L
FDA9! A5 3D              LDA     A1H
FDAB! 85 3F              STA     A2H
FDAD! A5 3C      MOD8CHK LDA     A1L
FDAF! 29 0F              and     #0F
FDB1! D0**               BNE     DATAOUT
FDB3! 20 92FD    XAM     JSR     PRA1
FDB1* 03
FDB6! A9 A0      DATAOUT LDA     #0A0
FDB8! 20 ****            JSR     COUT
FDBB! B1 3C              lda     @all,y
FDBD! 20 ****            jsr     prbyte
FDC0! 20 BAFC            jsr     nxtal
FDC3! 90E8               bcc     mod8chk
FDC5! 60                 RTS
FDC6!
FDC6! AD F904    sw7     lda     switch
FDC9! F0**               beq     sw740
FDCB! A5 20              lda     wndlft
FDCD! 4A                 lsr     a
FDCE! 60                 rts
FDC9* 04
FDCF! A9 28      sw740   lda     #028
FDD1! C5 21              cmp     wndwdth
FDD3! B0**               bcs     wdthok
FDD5! 85 21              sta     wndwdth
FDD3* 02
FDD7! A5 20      wdthok  lda     wndlft
FDD9! 60                 rts
FDDA!
FDBE* DAFD
FAF7* DAFD
FDDA! 48         PRBYTE  PHA
FDDB! 4A                 LSR     A
FDDC! 4A                 LSR     A
FDDD! 4A                 LSR     A
FDDE! 4A                 LSR     A
FDDF! 20 ****            JSR     PRHEXZ
FDE2! 68                 PLA
FDE3! 29 0F      PRHEX   AND     #0F
FDE0* E5FD
FDE5! 09 B0      PRHEXZ  ORA     #0B0
```

```
FDE7| C9 BA                  CMP   #0BA    ; ":"
FDE9| 90**                   BCC   COUT
FDEB| 69 06                  ADC   #6      ; ":".."?" -> "A".."F"
FDED|
FDED|                        .org  0FDED
FDE9* 02
FDB9* EDFD
FDA1* EDFD
FD90* 5B
FD6D* EDFD
FD65* EDFD
FD48* EDFD
FAF2* EDFD
FAED* EDFD
FAE7* EDFD
FDED| 6C 3600         COUT   JMP   @CSWL
FDF0| 48              COUT1  PHA
FDF1| 84 35                  STY   YSAV1
FDF3| 20 78FB                JSR   VIDWAIT
FDF6| A4 35                  LDY   YSAV1
FDF8| 68                     PLA
FDF9| 60                     RTS
FDFA|
FD97* FAFD
FDFA| 20 8EFD         newln  jsr   crout
FDFD| A9 A0                  lda   #0A0
FDFF| D0EC                   bne   cout
FE01|
FE01|
FE01|                        .include rom3
```

```
FE01!                          .page
FE01!                    ;
FE01!                    ;     monitor command page
FE01!                    ;
FE01!                          .org    0FE01
FE01! C6 34              BL1   DEC     YSAV
FE03! F09E                     BEQ     XAM8
FE05! CA                 BLANK DEX
FE06! D0**                     BNE     SETMDZ
FE08! C9 BA                    CMP     #0BA        ; ":"
FE0A! D0A7                     BNE     XAM
FE0C! 85 31              STOR  STA     MODE
FE0E! A5 3E                    LDA     A2L
FE10! 91 40                    STA     @A3L, Y
FE12! E6 40                    INC     A3L
FE14! D0**                     BNE     $1
FE16! E6 41                    INC     A3H
FE14* 02
FE18! 60                 $1    RTS
FE19!
FE19! A4 34              SETMODE LDY   YSAV
FE1B! B9 FF01                  LDA     IN-1, Y
FE06* 16
FE1E! 85 31              SETMDZ STA    MODE
FE20! 60                       RTS
FE21!
FE21! A2 01              LT    LDX     #1
FE23! B5 3E              $1    LDA     A2L, X
FE25! 95 42                    STA     A4L, X
FE27! CA                       DEX
FE28! 10F9                     BPL     $1
FE2A! 60                       RTS
FE2B!
FE2B! 00                       .org    0FE2C
FE2C! B1 3C              MOVE  LDA     @A1L, Y
FE2E! 91 42                    STA     @A4L, Y
FE30! 20 B4FC                  JSR     NXTA4
FE33! 90F7                     BCC     MOVE
FE35! 60                       RTS
FE36!
FE36! B1 3C              verify LDA    @A1L, Y
FE38! D1 42                    CMP     @A4L, Y
FE3A! F0**                     BEQ     $1
FE3C! 20 92FD                  JSR     PRA1
FE3F! B1 3C                    LDA     @A1L, Y
FE41! 20 DAFD                  JSR     PRBYTE
FE44! A9 BC                    LDA     #0BC    ; "<"
FE46! 20 EDFD                  JSR     COUT
FE49! A9 BE                    LDA     #0BE    ; ">"
FE4B! 20 EDFD                  JSR     COUT
FE4E! B1 42                    LDA     @A4L, Y
FE50! 20 DAFD                  JSR     PRBYTE
FE3A* 17
FE53! 20 B4FC            $1    JSR     NXTA4
```

```
FE56: 90DE                   BCC     verify
FE58: 60                     RTS
FE59:
FE59:                        .org    0FE59
FE59: 6C F203        BASCONT JMP     @softev
FE5C: 4C 00E0        XBASIC  JMP     BASIC
FE5F:
FE5F: 00                     .org    0FE60
FE60: 20 ****        LIST    jsr     a1pc
FE63: 20 D0F8        $1      jsr     instdsp
FE66: 20 53F9                jsr     pcadj
FE69: 85 3A                  sta     pcl
FE6B: 84 3B                  sty     pch
FE6D: C5 3E                  cmp     a2l
FE6F: 98                     tya
FE70: E5 3F                  sbc     a2h
FE72: 90EF                   bcc     $1
FE74: 60                     rts
FE75:
FE75:                        .org    0FE75
FE61* 75FE
FE75: 8A             A1PC    TXA
FE76: F0**                   BEQ     $2
FE78: B5 3C          $1      LDA     A1L, X
FE7A: 95 3A                  STA     PCL, X
FE7C: CA                     DEX
FE7D: 10F9                   BPL     $1
FE76* 07
FE7F: 60             $2      RTS
FE80:
FE80: A0 7F          SETINV  LDY     #7F
FE82: D0**                   BNE     SETIFLG
FE84: A0 FF          SETNORM LDY     #0FF
FE82* 02
FE86: 84 32          SETIFLG STY     INVFLG
FE88: 60                     RTS
FE89:
FE89: A9 00          SETKBD  LDA     #0
FE8B: 85 3E          INPORT  STA     A2L      ; IN#n
FE8D: A2 38          INPRT   LDX     #KSWL
FE8F: A0 1B                  LDY     #1B
FE91: D0**                   BNE     IOPRT
FE93: A9 00          SETVID  LDA     #0
FE95: 85 3E          OUTPORT STA     A2L      ; PR#n
FE97: A2 36          OUTPRT  LDX     #CSWL
FE99: A0 F0                  LDY     #0F0
FE91* 08
FE9B: A5 3E          IOPRT   LDA     A2L
FE9D: 29 07                  AND     #07      ; only slots 1..7 are legal
FE9F: F0**                   BEQ     IOPRT1   ; slot 0 has no I/O ROM space
FEA1: 09 C0                  ORA     #iopage
FEA3: A0 00                  LDY     #0
FEA5: F0**                   BEQ     IOPRT2
FE9F* 06
FEA7: A9 FD          IOPRT1  LDA     #0FD
FEA5* 02
```

```
FEA9: 94 00           IOPRT2  STY    loc0, X
FEAB: 95 01                   STA    loc1, X
FEAD: A5 3E                   lda    a21     ; if slot in [8..15.] then entry:=Cs08
FEAF: 29 08                   and    #8      ;                     else entry:=Cs00
FEB1: 15 00                   ora    loc0,x
FEB3: 95 00                   sta    loc0,x
FEB5: 60                      rts
FEB6:
FEB6:                         .org   0FEB6
FEB6: 20 75FE         GO      JSR    A1PC
FEB9: 20 ****                 JSR    RESTORE
FEBC: 6C 3A00                 JMP    @PCL
FEBF: 4C D7FA         REGZ    JMP    REGDSP
FEC2:
FEC2: 20 47F8         scrn80  jsr    gbascalc
FEC5: 4C ****                 jmp    scrn802
FEC8:
FEC8: 00 00                   .org   0FECA
FECA: 4C F803         USR     JMP    USRADR
```

```
FECD:                           .page
FECD:
FECD: 60                 write  rts     ; no tape  out!
FECE:
FECE: 08                 stor80 php
FECF: A4 24                     ldy     ch
FED1: 20 C9FC                   jsr     selbnk
FED4: 4C ****                   jmp     strts
FED7:
FD27* D7FE
FD16* D7FE
FED7: 08                 curs80 php
FED8: A4 24                     ldy     ch
FEDA: 20 C9FC                   jsr     selbnk
FEDD: B1 28                     lda     @basl,y
FEDF: 49 80                     eor     #bit7
FED5* E1FE
FEE1: 91 28              strts  sta     @basl,y ; write char,
FCF8* E3FE
FEE3: AC 7904            vidrts ldy     chy     ; restore Yreg,
FCEB* E6FE
FEE6: 8D 0CC0            vidplp sta     vidbnk  ; restore memory bank,
FEE9: 28                        plp             ; restore Iflag
FEEA: 60                        rts
FEEB:
FC28* EBFE
FEEB: 20 C6FD            vtab80 jsr     sw7
FEEE: 18                        clc
FEEF: 65 28                     adc     basl
FEF1: 85 28                     sta     basl
FEF3: 60                        rts
FEF4:
FEF4:
FEF4: 00 00                     .org    0FEF6
FEF6: 20 01FE            CRMON  JSR BL1
FEF9: 68                        PLA
FEFA: 68                        PLA
FEFB: D0**                      BNE MONZ
FEFD:
FEFD: 60                 read   rts     ; no tape input!
FEFE:
FD7D* FEFE
FEFE: 08                 get80  php
FEFF: A4 24                     ldy     ch
FF01: 20 C9FC                   jsr     selbnk
FF04: B1 28                     lda     @basl,y
FF06: 4C E3FE                   jmp     vidrts
FF09:                    ;
FF09:                    ; fast scroll line without jsr selbnk
FF09:                    ;
FF09: 08                 scrl80 php
FF0A: 78                        sei                     ; DANGER: 400..BFF is switched!
FF0B: 4A                        lsr     A
FF0C: A8                        tay
```

```
FF0D: 90**                     bcc    evenchr        ; first time odd or even?
FF0F: 8D 0DC0         oddchr  sta    vidbnk+1       ; static RAM on
FF12: B1 28                   lda    @basl,y        ; copy in static RAM
FF14: 91 2A                   sta    @bas2l,y
FF16: 8D 0CC0                 sta    vidbnk         ; static RAM off
FF19: CE 7904                 dec    chy
FF1C: 30**                    bmi    scrlex  ; ready?
FF0D* 0F
FF1E: B1 28           evenchr lda    @basl,y        ; copy in dynamic RAM
FF20: 91 2A                   sta    @bas2l,y
FF22: 88                      dey
FF23: CE 7904                 dec    chy
FF26: 10E7                    bpl    oddchr  ; more to scroll?
FF1C* 0A
FF28: 28              scrlex  plp
FF29: 18                      clc
FF2A: 60                      rts
FF2B:
FF2B: 00 00                   .org   0FF2D
FF2D: 60              PRERR    rts
FF2E:
FEC6* 2EFF
FF2E: 20 C9FC         scrn802 jsr    selbnk
FF31: B1 26                   lda    @gbasl,y
FF33: 8D 0CC0                 sta    vidbnk
FF36: AC 7904                 ldy    chy
FF39: 60                      rts
FF3A:
FD5D* 3AFF
FF3A: A9 87           BELL     LDA   #87
FF3C: 4C EDFD                  JMP   COUT
```

```
FF3F:                          .page
FEBA* 3FFF
FF3F: A5 48          RESTORE  LDA    STATUS
FF41: 48                      PHA
FF42: A5 45                   LDA    acc
FF44: A6 46          RESTR1   LDX    Xreg
FF46: A4 47                   LDY    Yreg
FF48: 28                      PLP
FF49: 60                      RTS
FF4A:
FF4A: 85 45          SAVE     STA    acc
FF4C: 86 46          SAV1     STX    Xreg
FF4E: 84 47                   STY    Yreg
FF50: 08                      PHP
FF51: 68                      PLA
FF52: 85 48                   STA    status
FF54: BA                      TSX
FF55: 86 49                   STX    spnt     ; save the wrong stack pointer value!
FF57: D8                      CLD    .
FF58:                         .org   0FF58
FF58: 60             iorts    RTS             ; used by slot ROM
FF59:
```

```
FF59!                          .page
FF59! 20 84FE           OLDRST  JSR     SETNORM
FF5C! 20 2FFB                   JSR     INIT
FF5F! 20 93FE                   JSR     SETVID
FF62! 20 89FE                   JSR     SETKBD
FF65!
FF65! D8                MON     CLD
FF66! 20 3AFF                   JSR     BELL
FEFB* 6C
FF69! A9 AA             MONZ    LDA     #0AA    ; "*"
FF6B! 85 33                     STA     PROMPT
FF6D! 20 67FD                   JSR     GETLNZ
FF70! 20 ****                   JSR     ZMODE
FF73! 20 ****           NXTITM  JSR     GETNUM
FF76! 84 34                     STY     YSAV
FF78! A0 11                     LDY     #011
FF7A! 88                CHRSRCH DEY
FF7B! 30E8                      BMI     MON
FF7D! D9 ****                   CMP     CHRTBL, Y
FF80! D0F8                      BNE     CHRSRCH
FF82! 20 ****                   JSR     TOSUB
FF85! A4 34                     LDY     YSAV
FF87! 4C 73FF                   JMP     NXTITM
FF8A! A2 03             DIG     LDX     #3
FF8C! 0A                        ASL     A
FF8D! 0A                        ASL     A
FF8E! 0A                        ASL     A
FF8F! 0A                        ASL     A
FF90! 0A                NXTBIT  ASL     A
FF91! 26 3E                     ROL     A2L
FF93! 26 3F                     ROL     A2H
FF95! CA                        DEX
FF96! 10F8                      BPL     NXTBIT
FF98! A5 31             NXTBAS  LDA     MODE
FF9A! D0**                      BNE     NXTBS2
FF9C! B5 3F                     LDA     A2H, X
FF9E! 95 3D                     STA     A1H, X
FFA0! 95 41                     STA     A3H, X
FF9A* 06
FFA2! E8                NXTBS2  INX
FFA3! F0F3                      BEQ     NXTBAS
FFA5! D0**                      BNE     NXTCHR
FF74* A7FF
FFA7! A2 00             GETNUM  LDX     #0
FFA9! 86 3E                     STX     A2L
FFAB! 86 3F                     STX     A2H
FFA5* 06
FFAD! 20 2BFC           NXTCHR  jsr     getupcs
FFB0! 49 B0                     EOR     #0B0
FFB2! C9 0A                     CMP     #0A
FFB4! 90D4                      BCC     DIG
FFB6! 69 88                     ADC     #88
FFB8! C9 FA                     CMP     #0FA
FFBA! B0CE                      BCS     DIG
```

```
FFBC: 60                       RTS
FFBD:
FFBD: 00                       .org    0FFBE
FF83* BEFF
FFBE: A9 FE          TOSUB     LDA     #0FE     ; command page
FFC0: 48                       PHA
FFC1: B9 ****                  LDA     SUBTBL, Y
FFC4: 48                       PHA             ; JMP by RTS
FFC5: A5 31                    LDA     MODE
FF71* C7FF
FFC7: A0 00          ZMODE     LDY     #0
FFC9: 84 31                    STY     MODE
FFCB: 60                       RTS
```

```
FFCC|                          .page
FFCC|                          .org    0FFCC
FF7E* CCFF
FFCC| EA              CHRTBL  .byte   0EA       ;  Q
FFCD| BB                      .byte   0BB       ; ctrl-B
FFCE| EE                      .byte   0EE       ;  U
FFCF| 98                      .byte   098       ;  ?
FFD0| EF                      .byte   0EF       ;  V
FFD1| 06                      .byte   006       ;  M
FFD2| 04                      .byte   004       ;  K
FFD3| E9                      .byte   0E9       ;  P
FFD4| 07                      .byte   007       ;  N
FFD5| 02                      .byte   002       ;  I
FFD6| 05                      .byte   005       ;  L
FFD7| 00                      .byte   000       ;  G
FFD8| 93                      .byte   093       ;  :
FFD9| A7                      .byte   0A7       ;  .
FFDA| 95                      .byte   095       ;  <
FFDB| C6                      .byte   0C6       ; ctrl-M
FFDC| 99                      .byte   099       ;  blank
FFDD|
FFC2* DDFF
FFDD| 58              SUBTBL  .byte   058       ; Basic warm    jmp 03F2  is move
FFDE| 5B                      .byte   05B       ; Basic cold    jmp 0E000 is move
FFDF| C9                      .byte   0C9       ; user          jmp 03F8
FFE0| BE                      .byte   0BE       ; register display
FFE1| 35                      .byte   035       ; verify
FFE2| 2B                      .byte   02B       ; move
FFE3| 8C                      .byte   08C       ; input vector
FFE4| 96                      .byte   096       ; output vector
FFE5| 83                      .byte   083       ; normal
FFE6| 7F                      .byte   07F       ; inverse
FFE7| 5F                      .byte   05F       ; list  is moved!
FFE8| B5                      .byte   0B5       ; go
FFE9| 18                      .byte   018       ; :
FFEA| 18                      .byte   018       ; .
FFEB| 20                      .byte   020       ; <
FFEC| F5                      .byte   0F5       ; <cr>
FFED| 04                      .byte   004       ; <space>
FFEE|
FFEE| 84 2D           clrsc3  sty     v2
FFF0| A0 4F                   ldy     #04f      ; 80-col -1
FFF2| AD F904                 lda     switch
FFF5| D0**                    bne     clr80
FFF7| A0 27                   ldy     #027      ; 40-col -1
FFF5* 02
FFF9| 60              clr80   rts
FFFA|
FFFA|                         .org     0FFFA
FFFA| FB03                    .word    NMI
FFFC| 62FA                    .word    RESET
FFFE| 40FA                    .word    IRQ
0000|
0000|                              .end
```

SYMBOLTABLE DUMP

```
AB - Absolute     LB - Label     UD - Undefined     MC - Macro
RF - Ref          DF - Def       PR - Proc          FC - Func
PB - Public       PV - Private   CS - Consts


A1H      AB 003D¦ A1L      AB 003C¦ A1PC     LB FE75¦
A2H      AB 003F¦ A2L      AB 003E¦ A3H      AB 0041¦
A3L      AB 0040¦ A4H      AB 0043¦ A4L      AB 0042¦
ACC      AB 0045¦ ADDINP   LB FD84¦ ADVANCE  LB FBF4¦
AMPERV   AB 03F5¦ BAS2H    AB 002B¦ BAS2L    AB 002A¦
BASCALC  LB FBC1¦ BASCONT  LB FE59¦ BASH     AB 0029¦
BASIC    AB E000¦ BASIC2   AB E003¦ BASL     AB 0028¦
BCKSPC   LB FD71¦ BELL     LB FF3A¦ BELL1    LB FBD9¦
BIT7     AB 0080¦ BL1      LB FE01¦ BLANK    LB FE05¦
BREAK    LB FA4C¦ BRKV     AB 03F0¦ BS       LB FC10¦
CANCEL   LB FD62¦ CH       AB 0024¦ CHAR1    LB F9B4¦
CHAR2    LB F9BA¦ CHRBAS   AB C006¦ CHRFONT  AB 0001¦
CHRGEN0  AB C002¦ CHRGEN1  AB C004¦ CHRINV   AB C000¦
CHRSRCH  LB FF7A¦ CHRTBL   LB FFCC¦ CHY      AB 0479¦
CLEOL80  LB FCDB¦ CLEOLZ   LB FC9E¦ CLEOP1   LB FC46¦
CLR80    LB FFF9¦ CLREOL   LB FC9C¦ CLREOP   LB FC42¦
CLRROM   AB CFFF¦ CLRSC2   LB F838¦ CLRSC3   LB FFEE¦
CLRSCR   LB F832¦ CLRTOP   LB F836¦ COLOR    AB 0030¦
COUT     LB FDED¦ COUT1    LB FDF0¦ CR       LB FC62¦
CRMON    LB FEF6¦ CROUT    LB FD8E¦ CSWH     AB 0037¦
CSWL     AB 0036¦ CURS80   LB FED7¦ CV       AB 0025¦
DATAOUT  LB FDB6¦ DIG      LB FF8A¦ DISKID   LB FB01¦
ERR      LB F8A5¦ EVENCHR  LB FF1E¦ FIXSEV   LB FA98¦
FMT1     LB F962¦ FMT2     LB F9A6¦ FORMAT   AB 002E¦
GBASCALC LB F847¦ GBASH    AB 0027¦ GBASL    AB 0026¦
GET80    LB FEFE¦ GETFMT   LB F8A9¦ GETLN    LB FD6A¦
GETLNZ   LB FD67¦ GETNUM   LB FFA7¦ GETUPCS  LB FC2B¦
GO       LB FEB6¦ H2       AB 002C¦ HLINE    LB F819¦
HOME     LB FC58¦ IEVEN    LB F89B¦ IN       AB 0200¦
INIT     LB FB2F¦ INPORT   LB FE8B¦ INPRT    LB FE8D¦
INSDS1   LB F882¦ INSDS2   LB F88C¦ INSTDSP  LB F8D0¦
INVFLG   AB 0032¦ IOARD    AB C000¦ IOPAGE   AB 00C0¦
IOPRT    LB FE9B¦ IOPRT1   LB FEA7¦ IOPRT2   LB FEA9¦
IORTS    LB FF58¦ IRQ      LB FA40¦ IRQLOC   AB 03FE¦
JLOCAL   LB FBB6¦ KBD      AB C000¦ KBDEXTN  AB C008¦
KBDSTRB  AB C010¦ KEYIN    LB FD1B¦ KSWH     AB 0039¦
KSWL     AB 0038¦ LASTIN   AB 002F¦ LENGTH   AB 002F¦
LF       LB FC66¦ LINE1    AB 0400¦ LIST     LB FE60¦
LMNEM    AB 002C¦ LOC0     AB 0000¦ LOC1     AB 0001¦
LOCAL    LB FBA0¦ LOCCHR   LB FB09¦ LOCJMP   LB FB99¦
```

```
LOGO     LB FB60¦ LOGO1    LB FD0F¦ LORES    AB C056¦
LOWSCR   AB C054¦ LT       LB FE21¦ MASK     AB 002E¦
MIXCLR   AB C052¦ MNEML    LB F9C0¦ MNEMR    LB FA00¦
MNNDX1   LB F8BE¦ MNNDX2   LB F8C2¦ MNNDX3   LB F8C9¦
MOD8CHK  LB FDAD¦ MODE     AB 0031¦ MON      LB FF65¦
MONZ     LB FF69¦ MOVE     LB FE2C¦ MSLOT    AB 07F8¦
NEWLN    LB FDFA¦ NEWMON   LB FA6F¦ NMI      AB 03FB¦
NOCTRL   LB FBED¦ NOFIX    LB FAA0¦ NOTCR    LB FD3D¦
NOTCR1   LB FD5F¦ NXTA1    LB FCBA¦ NXTA4    LB FCB4¦
NXTBAS   LB FF98¦ NXTBIT   LB FF90¦ NXTBS2   LB FFA2¦
NXTCHAR  LB FD75¦ NXTCHR   LB FFAD¦ NXTCOL   LB F85F¦
NXTITM   LB FF73¦ ODDCHR   LB FF0F¦ OLDBRK   LB FA59¦
OLDRST   LB FF59¦ OUTPORT  LB FE95¦ OUTPRT   LB FE97¦
PADDL0   AB C064¦ PCADJ    LB F953¦ PCADJ2   LB F954¦
PCADJ3   LB F956¦ PCADJ4   LB F95C¦ PCH      AB 003B¦
PCL      AB 003A¦ PIP      LB FBDF¦ PLOT     LB F800¦
PLOT1    LB F80E¦ PLOT80   LB FCED¦ PRA1     LB FD92¦
PRADR1   LB F910¦ PRADR2   LB F914¦ PRADR4   LB F92A¦
PRADR5   LB F930¦ PRBL2    LB F94A¦ PRBL3    LB F94C¦
PRBLNK   LB F948¦ PRBYTE   LB FDDA¦ PREAD    LB FB1E¦
PRERR    LB FF2D¦ PRHEX    LB FDE3¦ PRHEXZ   LB FDE5¦
PRNTAX   LB F941¦ PRNTBL   LB F8DB¦ PRNTOP   LB F8D4¦
PRNTX    LB F944¦ PRNTYX   LB F940¦ PROMPT   AB 0033¦
PRYX2    LB FD96¦ PTRIG    AB C070¦ PWRCON   LB FAFD¦
PWREDUP  AB 03F4¦ PWRUP    LB FAA3¦ RDCHAR   LB FD35¦
RDCHAR1  LB FBB9¦ RDKEY    LB FD0C¦ RDKEY2   LB FD15¦
READ     LB FEFD¦ REGDSP   LB FAD7¦ REGZ     LB FEBF¦
RELADR   LB F938¦ RESET    LB FA62¦ RESTORE  LB FF3F¦
RESTR1   LB FF44¦ RGDSP1   LB FADA¦ RMNEM    AB 002D¦
RNDH     AB 004F¦ RNDL     AB 004E¦ RTBL     LB FB19¦
RTS1     LB F831¦ RTS2     LB F961¦ RTS4     LB FC2A¦
SAV1     LB FF4C¦ SAVE     LB FF4A¦ SCRL80   LB FF09¦
SCRLEX   LB FF28¦ SCRN     LB F871¦ SCRN2    LB F879¦
SCRN80   LB FEC2¦ SCRN802  LB FF2E¦ SCROLL   LB FC70¦
SELBNK   LB FCC9¦ SELBNK2  LB FCCC¦ SETCOL   LB F864¦
SETGR    LB FB40¦ SETIFLG  LB FE86¦ SETINV   LB FE80¦
SETKBD   LB FE89¦ SETMDZ   LB FE1E¦ SETMODE  LB FE19¦
SETNORM  LB FE84¦ SETPG3   LB FAAA¦ SETPLP   LB FAAC¦
SETPWRC  LB FB6F¦ SETTXT   LB FB39¦ SETVID   LB FE93¦
SETWND   LB FB4B¦ SLOOP    LB FABB¦ SOFTEV   AB 03F2¦
SPKR     AB C030¦ SPNT     AB 0049¦ STATUS   AB 0048¦
STOR     LB FE0C¦ STOR80   LB FECE¦ STORADV  LB FBF0¦
STORINV  LB FBEE¦ STRTS    LB FEE1¦ SUBTBL   LB FFDD¦
SW1      LB FB10¦ SW2      LB FB13¦ SW3      LB FCFA¦
SW4      LB FD03¦ SW5      LB FC36¦ SW6      LB FC37¦
SW7      LB FDC6¦ SW740    LB FDCF¦ SWITCH   AB 04F9¦
TABV     LB FB5B¦ TAPEIN   AB C060¦ TAPEOUT  AB C020¦
TITLE    LB FB90¦ TOSUB    LB FFBE¦ TTLOUT0  AB C058¦
TTLOUT1  AB C05A¦ TTLOUT2  AB C05C¦ TTLOUT3  AB C05E¦
TXTCLR   AB C050¦ UP       LB FC1A¦ UPPER    LB FC2F¦
```

MONITOR   FILE:ROM3.TEXT

```
USR       LB FECA¦  USRADR   AB 03F8¦  V2        AB 002D¦
VERIFY    LB FE36¦  VID40    AB C00A¦  VID80     AB C00B¦
VIDBNK    AB C00C¦  VIDOUT   LB FBFD¦  VIDPLP    LB FEE6¦
VIDRTS    LB FEE3¦  VIDWAIT  LB FB78¦  VLINE     LB F828¦
VLINEZ    LB F826¦  VTAB     LB FC22¦  VTAB80    LB FEEB¦
VTABZ     LB FC24¦  WAIT     LB FCA8¦  WDTHOK    LB FDD7¦
WIDTH     AB 0050¦  WNDBTM   AB 0023¦  WNDLFT    AB 0020¦
WNDTOP    AB 0022¦  WNDWDTH  AB 0021¦  WRITE     LB FECD¦
XAM       LB FDB3¦  XAM8     LB FDA3¦  XBASIC    LB FE5C¦
XREG      AB 0046¦  YREG     AB 0047¦  YSAV      AB 0034¦
YSAV1     AB 0035¦  ZMODE    LB FFC7¦                   ¦
```

```
Current minimum space is 6038 words
F810* C9FC
F813* EDFC
F887* 96FD
F946* DAFD
F942* DAFD
F92E* DAFD
F8D7* DAFD
F94D* EDFD
F924* EDFD
F91C* EDFD
F904* EDFD
FA64* 84FE
FA6D* 89FE
FA6A* 93FE
F874* C2FE
FBF1* CEFE
FB17* 09FF
FA71* 3AFF
FA4E* 4CFF
FA60* 65FF
F839* EEFF

Assembly complete:      1335 lines
     0   Errors flagged on this Assembly
```

```
0000:                      .absolute
0000:                      .proc   printer
Current memory available:   8644
0000:
0000: 0021          version .equ    21      ; version 2.1
0000:
0000: C100          rom     .equ    0C100
0000: 00C1          rompage .equ    0C1
0000:                       .org    rom
C100:
C100: C090          devsel  .equ    0C090
C100: C1C1          pready  .equ    0C1C1
C100:
C100: C090          preg    .equ    devsel
C100: C098          acia    .equ    devsel+8
C100:
C100: C098          inreg   .equ    acia+0  ;  7    6    5    4    3    2    1    0
C100: C098          outreg  .equ    acia+0  ;
C100:
C100: C099          stsreg  .equ    acia+1  ; IRQ  DSR  DCD  tran rec  ovr- frm- par-
C100:                                       ; occur inact inact empty full    e r r o r
C100:
C100: C09A          cmdreg  .equ    acia+2  ;   parity  par  rec  transmit-  rec-  DTR
C100:                                       ;  mode-ctrl  enabl echo  IRQ,RTS,brk IRQ  activ
C100:
C100: C09B          ctrlreg .equ    acia+3  ; 2 stop word-     clock  b a u d   r a t e
C100:                                       ; bits      length intrn
C100:
C100: 0478          Accu    .equ    478     ; save char
C100: 04F8          chanel  .equ    4F8     ; par/ser out switch : if chanel<80 then par else ser
C100:
C100: 0479          vid0    .equ    479     ; used in the 80-col screen driver
C100: 04F9          vid1    .equ    4F9     ; reserved
C100: 0579          vid2    .equ    579     ; reserved
C100: 05F9          modechk .equ    5F9     ; warmstart byte
C100: 0679          mode    .equ    679     ; CR->CR/LF   video echo
C100:                                       ; par   ser   par  ser
C100: 06F9          ctrl    .equ    6F9     ; value for ACIA ctrl-reg
C100: 0779          cmd     .equ    779     ; value for ACIA cmd-reg
C100:
C100: 07F9          hCount  .equ    7F9
C100:
C100: 0024          ch      .equ    24
C100: 0036          csw     .equ    36
C100: 0038          ksw     .equ    38
C100: FDF0          cout1   .equ    0FDF0
C100:
C100: 002C          bit_a   .equ    2C
C100: 20 ****               jsr     init
C103: 90**                  bcc     pwrite2
C105:
C105:                       .org    rom+5
C105: 48            byte5   pha             ; tested by Pascal
C106: 21                    .byte   version
C107: 48            byte7   pha             ; tested by Pascal
C108:
```

```
C108:                     .org    rom+8
C108: 48            v24    pha            ; first entry for IN#9 or PR#9
C109: A5 39                lda    Ksw+1
C10B: C9 C1                cmp    #rompage
C10D: D0**                 bne    swrite  ; no
C10F: 68                   pla
C110: A9 14         sread  lda    #14     ; yes, first entry
C112: 85 38                sta    Ksw     ; zap entry to sread2
C114:
C114:                      .org   rom+14
C114: 20 ****       sread2 jsr    init
C117: A9 08                lda    #8
C119: 2C 99C0       $0     bit    stsreg
C11C: F0FB                 beq    $0
C11E: AD 98C0              lda    inreg
C121: 49 80                eor    #80
C123: 60                   rts
C124:
C124:
C10D* 15
C124: A9 29         swrite lda    #29     ; first PR#9 entry
C126: 85 36                sta    csw     ; zap entry vector
C128: 68                   pla
C129:
C129:                      .org   rom+29
C129: 20 ****       swrite2 jsr   init    ; setup the 6551
C12C: 38                   sec
C12D:
C103* 28
C12D: 6E F804       pwrite2 ror   chanel
C130:               output
C130: EE F907       $1     inc    hCount
C133: A5 24                lda    ch
C135: CD F907              cmp    hCount
C138: 90**                 bcc    notab
C13A: A9 A0                lda    #0A0
C13C: 20 ****              jsr    out1
C13F: 4C 30C1              jmp    $1
C138* 08
C142: 20 ****       notab  jsr    out
C145: C9 0D                cmp    #0D
C147: D0**                 bne    nocr
C149: 20 ****              jsr    cCount
C14C: 2C 7906              bit    mode
C14F: 10**                 bpl    nocr
C151: A9 8A                lda    #8A
C153: 20 ****              jsr    out1
C14F* 05
C147* 0D
C156: 2C 7906       nocr   bit    mode
C159: AD 7804              lda    accu
C15C: 50**                 bvc    ret
C15E: 4C F0FD              jmp    cout1
C161:
C161: 00                   brk
```

```
C162!
C12A* 62C1
C115* 62C1
C101* 62C1
C162! 8D 7804          init   sta   Accu      ; low(addr)=Fx
C165! AD F905                 lda   modechk
C168! 49 A5                   eor   #0A5      ; printer/v24 warmstart?
C16A! CD 7906                 cmp   mode
C16D! F0**                    beq   warm      ; yes
C16F!
C16F! A9 9E                   lda   #9E       ; no, set default values: 8 data+2 stop bits,
C171! 8D F906                 sta   ctrl      ;                         9600 baud
C174!
C174! A9 0B                   lda   #0B       ; no parity, DTR=low,   RTS=low
C176! 8D 7907                 sta   cmd
C179!
C179! A9 C0                   lda   #0C0       ; mode bit 7:   CR->CR/LF translation on
C17B! 8D 7906                 sta   mode       :      bit 6:   output echo to video
C17E!
C17E! 49 A5                   eor   #0A5
C180! 8D F905                 sta   modechk  ; set warmstart flag
C183!
C14A* 83C1
C183! A9 00            cCount lda   #0
C185! 8D F907                 sta   hCount     init Tabulator count
C188!
C16D* 19
C188! AD 7907          warm   lda   cmd
C18B! CD 9AC0                 cmp   cmdreg   ; is the 6551 cmd  register ok ?
C18E! F0**                    beq   $1
C190! 8D 9AC0                 sta   cmdreg   ; no
C18E* 03
C193! AD F906          $1     lda   ctrl
C196! CD 9BC0                 cmp   ctrlreg  ; is the 6551 ctrl register ok ?
C199! F0**                    beq   $2
C19B! 8D 9BC0                 sta   ctrlreg  ; no
C199* 03
C19E! 18               $2     clc
C15C* 41
C19F! 60               ret    rts
C1A0!
C143* A0C1
C1A0! AD 7804          out    lda   Accu
C154* A3C1
C13D* A3C1
C1A3! 49 80            out1   eor   #80
C1A5! 2C F804                 bit   chanel
C1A8! 10**                    bpl   pout
C1AA!
C1AA! 48               sout   pha             ; save char
C1AB! A9 10                   lda   #10
C1AD! 2C 99C0          $0     bit   stsreg  ; ready for next char ?
C1B0! F0FB                    beq   $0      ; no, wait
C1B2! 68                      pla             ; yes
C1B3! 8D 98C0                 sta   outreg  ; send it
```

```
C1B6: 60                    rts
C1B7:
C1A8* 0D
C1B7: 2C C1C1         pout  bit    pready
C1BA: 30FB                  bmi    pout
C1BC: 8D 90C0               sta    devsel
C1BF: 60                    rts
C1C0:
C1C0:                       .org   rom+0C0
C1C0:                       .end
```

```
        PRINTER   FILE:PRINTER.21  SYMBOLTABLE DUMP


AB - Absolute     LB - Label     UD - Undefined    MC - Macro
RF - Ref          DF - Def       PR - Proc         FC - Func
PB - Public       PV - Private   CS - Consts


ACCU     AB 0478| ACIA     AB C098| BITA     AB 002C| BYTE5    LB C105|
BYTE7    LB C107| CCOUNT   LB C183| CH       AB 0024| CHANEL   AB 04F8|
CMD      AB 0779| CMDREG   AB C09A| COUT1    AB FDF0| CSW      AB 0036|
CTRL     AB 06F9| CTRLREG  AB C09B| DEVSEL   AB C090| HCOUNT   AB 07F9|
INIT     LB C162| INREG    AB C098| KSW      AB 0038| MODE     AB 0679|
MODECHK  AB 05F9| NOCR     LB C156| NOTAB    LB C142| OUT      LB C1A0|
OUT1     LB C1A3| OUTPUT   LB C130| OUTREG   AB C098| POUT     LB C1B7|
PREADY   AB C1C1| PREG     AB C090| PRINTER  PR ----| PWRITE2  LB C12D|
RET      LB C19F| ROM      AB C100| ROMPAGE  AB 00C1| SOUT     LB C1AA|
SREAD    LB C110| SREAD2   LB C114| STSREG   AB C099| SWRITE   LB C124|
SWRITE2  LB C129| V24      LB C108| VERSION  AB 0021| VID0     AB 0479|
VID1     AB 04F9| VID2     AB 0579| WARM     LB C188|


Current minimum space is 8231 words

Assembly complete:       158 lines
     0   Errors flagged on this Assembly
```

Stichwortverzeichnis

H

I

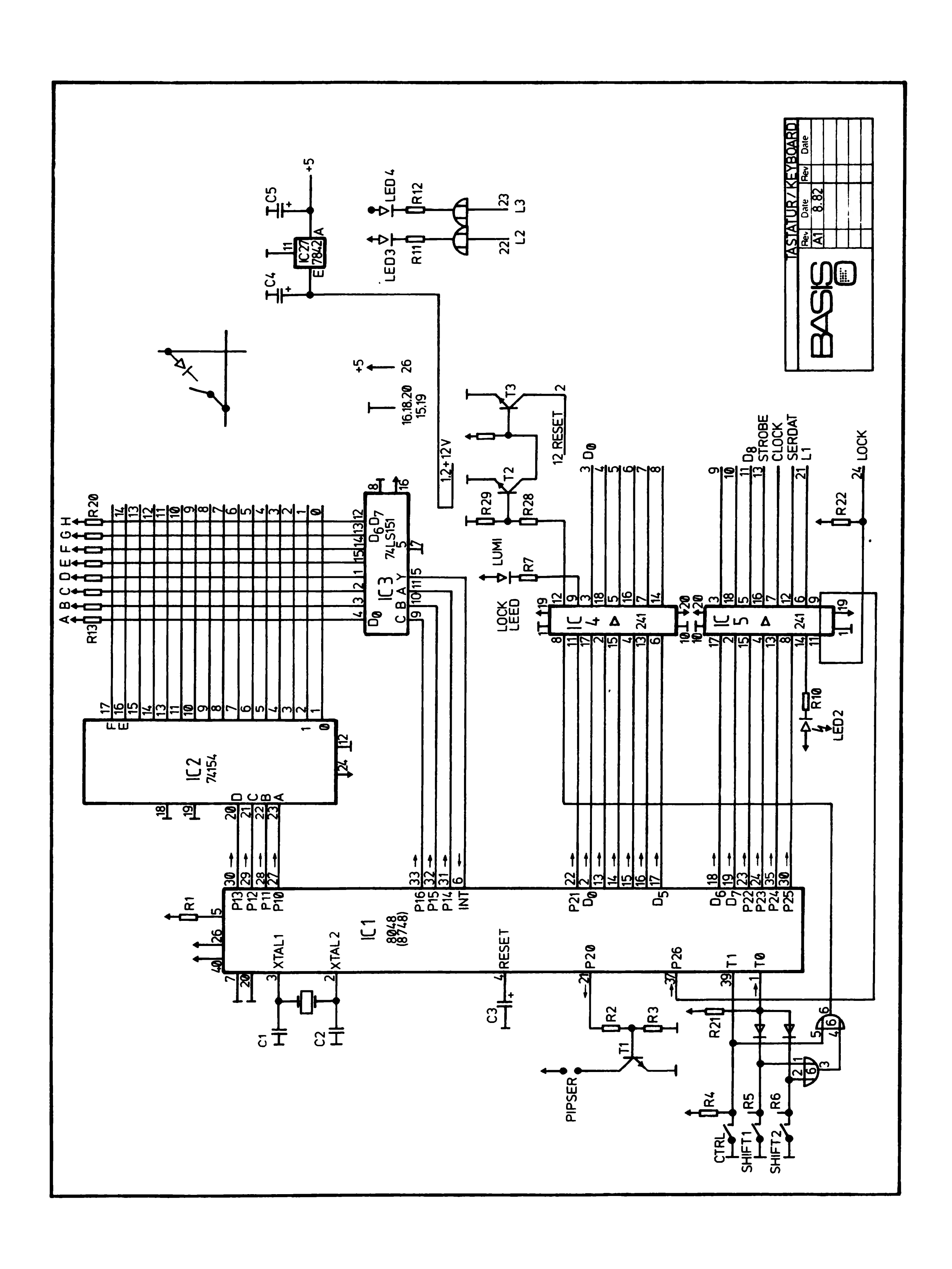
K

L

M

N

R



TASTATUR/KEYBOARD
BASIS
Rev
Date
A1
8.82
IC1
8048
(8748)
IC2
74154
IC3
74LS151
IC4
241
IC5
241
IC27
7842
RESET
XTAL1
XTAL2
LUMI
LOCK
LEED
STROBE
CLOCK
SERDAT
PIPSER
CTRL
SHIFT1
SHIFT2
LED2
LED3
LED4
+5
1,2+12V



D-4400 Münster
Postfach 1603
Telex 892 643 basis d
BTX 244